Acquisition Parameters of Graphs

Douglas B. West

Department of Mathematics
Zhejiang Normal University, Jinhua and
University of Illinois at Urbana-Champaign
west@math.uiuc.edu

Results with or by

Timothy D. LeSaulnier, Daniel C. McDonald,
Kevin G. Milans, Noah Prince, Chris Stocker,
Paul S. Wenger, Leslie Wiglesworth, Pratik Worah

The Problem

Model: People start with one vote. Some are friends.

A person can give his or her votes to a friend
if the friend has at least as many votes.

The Problem

Model: People start with one vote. Some are friends.

A person can give his or her votes to a friend
if the friend has at least as many votes.

Can one person acquire all the votes?

The Problem

Model: People start with one vote. Some are friends.

A person can give his or her votes to a friend
if the friend has at least as many votes.

Can one person acquire all the votes?

Model: (Wenger) Each city starts with one regiment.

The troops in one city can withdraw to a neighboring
city if that city already has at least as many troops.

The Problem

Model: People start with one vote. Some are friends.

A person can give his or her votes to a friend
if the friend has at least as many votes.

Can one person acquire all the votes?

Model: (Wenger) Each city starts with one regiment.

The troops in one city can withdraw to a neighboring
city if that city already has at least as many troops.

Can the troops be gathered in a single city?

The Problem

Model: People start with one vote. Some are friends.

A person can give his or her votes to a friend
if the friend has at least as many votes.

Can one person acquire all the votes?

Model: (Wenger) Each city starts with one regiment.

The troops in one city can withdraw to a neighboring city if that city already has at least as many troops.

Can the troops be gathered in a single city?

Def. total aquisition move = transfer all weight from u to a neighbor v; allowed if currently $w(u) \leq w(v)$.

The Problem

Model: People start with one vote. Some are friends.

A person can give his or her votes to a friend
if the friend has at least as many votes.

Can one person acquire all the votes?

Model: (Wenger) Each city starts with one regiment.

The troops in one city can withdraw to a neighboring
city if that city already has at least as many troops.

Can the troops be gathered in a single city?

Def. total aquisition move = transfer all weight from u to a neighbor v; allowed if currently $w(u) \leq w(v)$.

- End when $\{v\colon\ w(v) > 0\}$ is an independent set.

The Problem

Model: People start with one vote. Some are friends.

A person can give his or her votes to a friend
if the friend has at least as many votes.

Can one person acquire all the votes?

Model: (Wenger) Each city starts with one regiment.

The troops in one city can withdraw to a neighboring city if that city already has at least as many troops.

Can the troops be gathered in a single city?

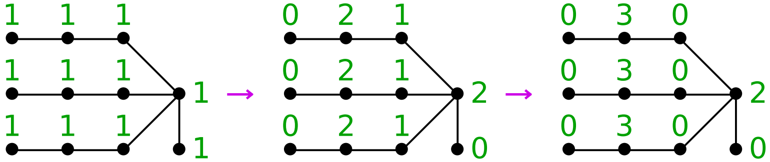
Def. total aquisition move = transfer all weight from u to a neighbor v; allowed if currently $w(u) \leq w(v)$.

- End when $\{v\colon\ w(v) > 0\}$ is an independent set.

Def. total aquisition number $a_t(G)$ = min size of the final indep. set when each vertex starts with weight 1.

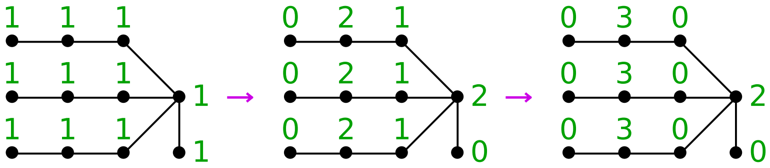
An Example

Ex. $a_t(T) = 4$, 11 vertices.



An Example

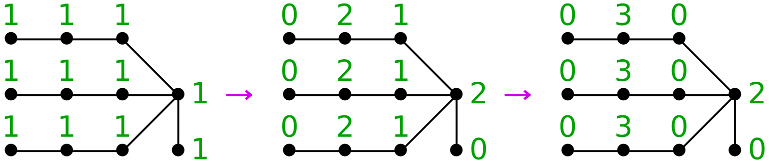
Ex. $a_t(T) = 4$, 11 vertices.



Alternative Models — Start with weight 1 on all.
Moving weight from u to v requires $w(v) \geq w(u)$.

An Example

Ex. $a_t(T) = 4$, 11 vertices.

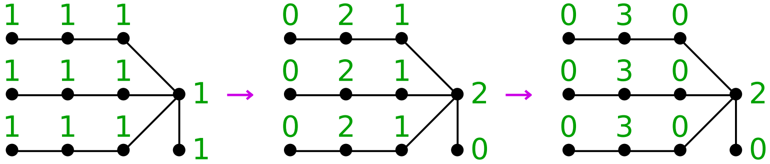


Alternative Models — Start with weight 1 on all.
Moving weight from u to v requires $w(v) \geq w(u)$.

total acquisition: move all weight from u — $a_t(G)$

An Example

Ex. $a_t(T) = 4$, 11 vertices.



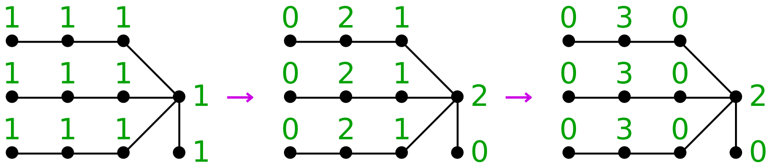
Alternative Models — Start with weight 1 on all.
Moving weight from u to v requires $w(v) \geq w(u)$.

total acquisition: move all weight from u — $a_t(G)$

unit acquisition: move any integer amount — $a_u(G)$

An Example

Ex. $a_t(T) = 4$, 11 vertices.



Alternative Models — Start with weight 1 on all.
Moving weight from u to v requires $w(v) \geq w(u)$.

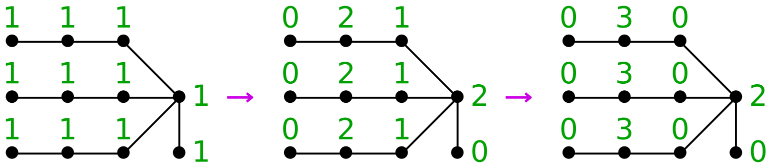
total acquisition: move all weight from u — $a_t(G)$

unit acquisition: move any integer amount — $a_u(G)$

fractional acquisition: move any positive amt — $a_f(G)$

An Example

Ex. $a_t(T) = 4$, 11 vertices.



Alternative Models — Start with weight 1 on all.
Moving weight from u to v requires $w(v) \geq w(u)$.

total acquisition: move all weight from u — $a_t(G)$

unit acquisition: move any integer amount — $a_u(G)$

fractional acquisition: move any positive amt — $a_f(G)$

game acquisition: move all weight, but two players Min and Max alternate moves — $a_g(G)$

Total Acquisition Number — Extremal Problem

All our graphs have n vertices.

Thm. (Lampert–Slater [1995]) If G is connected and nontrivial, then $a_t(G) \le \frac{n+1}{3}$, and this is sharp.

Total Acquisition Number — Extremal Problem

All our graphs have n vertices.

Thm. (Lampert–Slater [1995]) If G is connected and nontrivial, then $a_t(G) \leq \frac{n+1}{3}$, and this is sharp.

Pf. Since $H \subseteq G \Rightarrow a_t(H) \geq a_t(G)$,
it suffices to prove the bound for trees.

Total Acquisition Number — Extremal Problem

All our graphs have n vertices.

Thm. (Lampert–Slater [1995]) If G is connected and nontrivial, then $a_t(G) \le \frac{n+1}{3}$, and this is sharp.

Pf. Since $H \subseteq G \Rightarrow a_t(H) \ge a_t(G)$,
it suffices to prove the bound for trees.

Idea: Induction on n. Find a subtree T' with m vertices, $a_t(T') \le m/3$, and $T - V(T')$ connected.

Total Acquisition Number — Extremal Problem

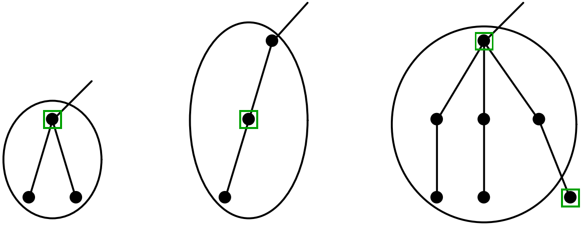
All our graphs have n vertices.

Thm. (Lampert–Slater [1995]) If G is connected and nontrivial, then $a_t(G) \le \frac{n+1}{3}$, and this is sharp.

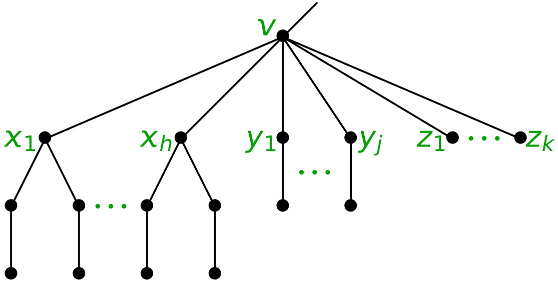
Pf. Since $H \subseteq G \Rightarrow a_t(H) \ge a_t(G)$,
it suffices to prove the bound for trees.

Idea: Induction on n. Find a subtree T' with m vertices, $a_t(T') \le m/3$, and $T - V(T')$ connected.

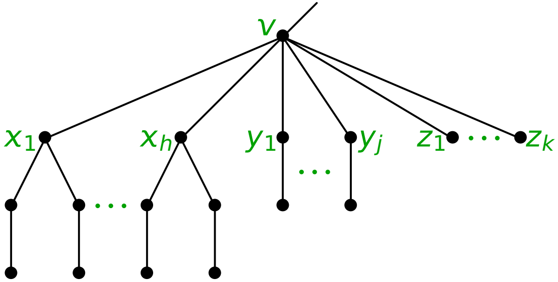
Easy cases:



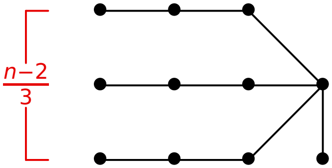
Hard Case:



Hard Case:



Sharpness:



Lower bound — An Obstruction

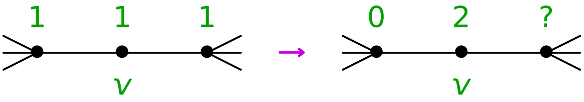
- View each initial unit of weight as a token or chip.

Lem. For total or unit acquisition, at most one chip can pass through a vertex v of degree 2, and it must be the chip from a neighbor.

Lower bound — An Obstruction

- View each initial unit of weight as a token or chip.

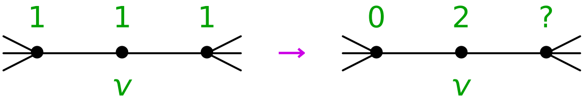
Lem. For total or unit acquisition, at most one chip can pass through a vertex v of degree 2, and it must be the chip from a neighbor.



Lower bound — An Obstruction

- View each initial unit of weight as a token or chip.

Lem. For total or unit acquisition, at most one chip can pass through a vertex v of degree 2, and it must be the chip from a neighbor.

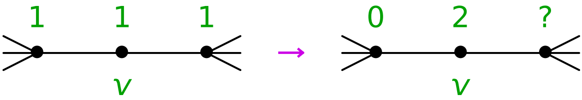


Cor. If the x, y-path in a tree has a vertex of degree 2 adjacent to neither x nor y, then the chips starting at x and y cannot combine.

Lower bound — An Obstruction

• View each initial unit of weight as a token or chip.

Lem. For total or unit acquisition, at most one chip can pass through a vertex v of degree 2, and it must be the chip from a neighbor.

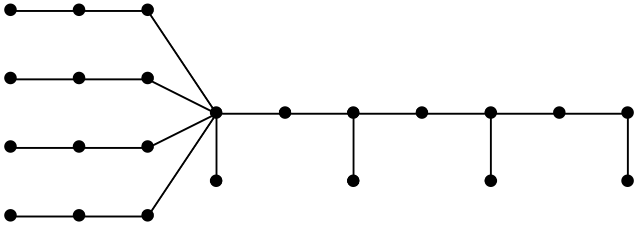


Cor. If the x, y-path in a tree has a vertex of degree 2 adjacent to neither x nor y, then the chips starting at x and y cannot combine.

• The lower bounds on a_t that use these observation apply also to a_u.

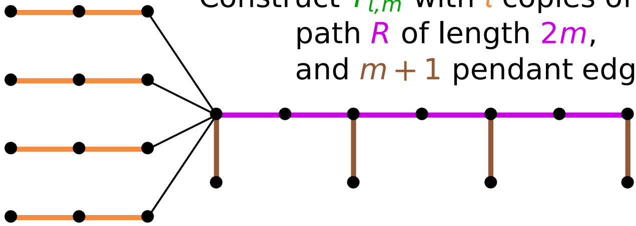
Trees with $a_t(G)$ Large

Ex. The tree $T_{l,m}$ with $(l, m) = (4, 3)$.



Trees with $a_t(G)$ Large

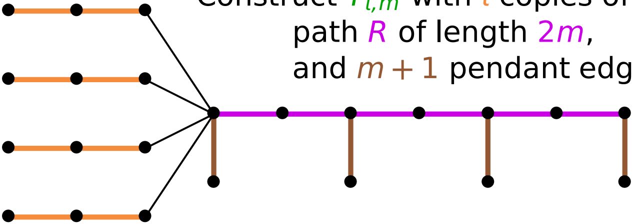
Ex. The tree $T_{l,m}$ with $(l, m) = (4, 3)$.



Construct $T_{l,m}$ with l copies of P_3,
path R of length $2m$,
and $m+1$ pendant edges.

Trees with $a_t(G)$ Large

Ex. The tree $T_{l,m}$ with $(l, m) = (4, 3)$.

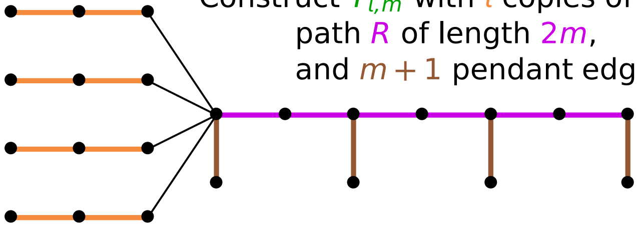


Construct $T_{l,m}$ with l copies of P_3,
path R of length $2m$,
and $m+1$ pendant edges.

#vertices = $3l + 3m + 2$, #leaves = $l + m + 1$
diameter = $2m + 4$, maxdegree = $l + 2$.

Trees with $a_t(G)$ Large

Ex. The tree $T_{l,m}$ with $(l, m) = (4, 3)$.



Construct $T_{l,m}$ with l copies of P_3, path R of length $2m$, and $m+1$ pendant edges.

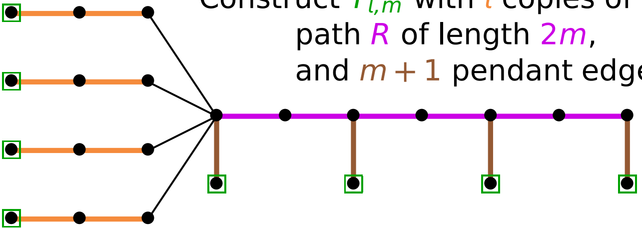
#vertices = $3l + 3m + 2$, #leaves = $l + m + 1$
diameter = $2m + 4$, maxdegree = $l + 2$.

Prop. (LPWWW [2013]) $a_t(T_{l,m}) = l + m + 1 = \frac{n+1}{3}$.

Pf. Chips from marked vertices cannot combine. ∎

Trees with $a_t(G)$ Large

Ex. The tree $T_{l,m}$ with $(l, m) = (4, 3)$.



Construct $T_{l,m}$ with l copies of P_3,
path R of length $2m$,
and $m + 1$ pendant edges.

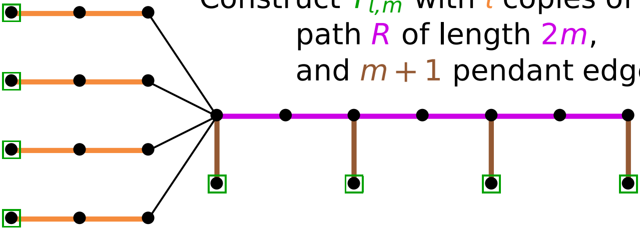
#vertices = $3l + 3m + 2$, #leaves = $l + m + 1$
diameter = $2m + 4$, maxdegree = $l + 2$.

Prop. (LPWWW [2013]) $a_t(T_{l,m}) = l + m + 1 = \frac{n+1}{3}$.

Pf. Chips from marked vertices cannot combine. ■

Trees with $a_t(G)$ Large

Ex. The tree $T_{l,m}$ with $(l, m) = (4, 3)$.



Construct $T_{l,m}$ with l copies of P_3,
path R of length $2m$,
and $m+1$ pendant edges.

#vertices = $3l + 3m + 2$, #leaves = $l + m + 1$
diameter = $2m + 4$, maxdegree = $l + 2$.

Prop. (LPWWW [2013]) $a_t(T_{l,m}) = l + m + 1 = \frac{n+1}{3}$.

Pf. Chips from marked vertices cannot combine. ■

Thm. For $d \geq 3$ and $k \geq 6$, there is a tree T with
$\Delta(T) = d$, $\mathrm{diam}\, T \geq k$, and $a_u(T) = a_t(T) = \frac{|V(T)|+1}{3}$.

The Extremal Graphs

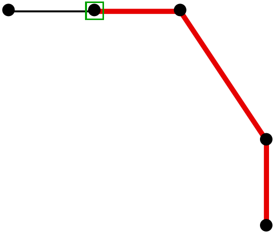
Thm. (LeSaulnier–West [2013]) An n-vertex graph G satisfies $a_t(G) = \frac{n+1}{3}$ if and only if G is a tree obtained from P_2 by iteratively growing a 3-edge path from a neighbor of a leaf.

The Extremal Graphs

Thm. (LeSaulnier–West [2013]) An n-vertex graph G satisfies $a_t(G) = \frac{n+1}{3}$ if and only if G is a tree obtained from P_2 by iteratively growing a 3-edge path from a neighbor of a leaf.

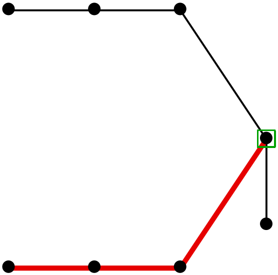
The Extremal Graphs

Thm. (LeSaulnier–West [2013]) An n-vertex graph G satisfies $a_t(G) = \frac{n+1}{3}$ if and only if G is a tree obtained from P_2 by iteratively growing a 3-edge path from a neighbor of a leaf.



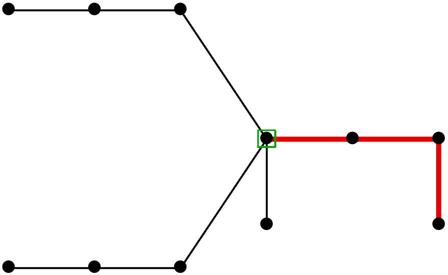
The Extremal Graphs

Thm. (LeSaulnier–West [2013]) An n-vertex graph G satisfies $a_t(G) = \frac{n+1}{3}$ if and only if G is a tree obtained from P_2 by iteratively growing a 3-edge path from a neighbor of a leaf.



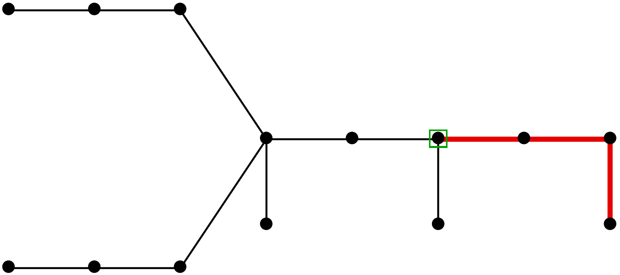
The Extremal Graphs

Thm. (LeSaulnier–West [2013]) An n-vertex graph G satisfies $a_t(G) = \frac{n+1}{3}$ if and only if G is a tree obtained from P_2 by iteratively growing a 3-edge path from a neighbor of a leaf.



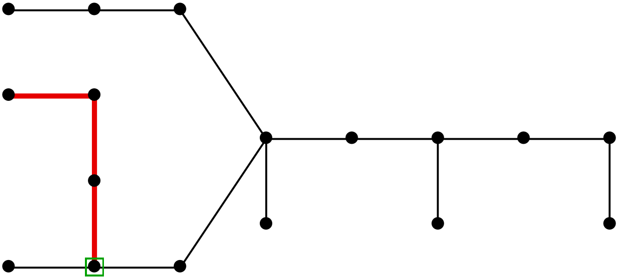
The Extremal Graphs

Thm. (LeSaulnier–West [2013]) An n-vertex graph G satisfies $a_t(G) = \frac{n+1}{3}$ if and only if G is a tree obtained from P_2 by iteratively growing a 3-edge path from a neighbor of a leaf.



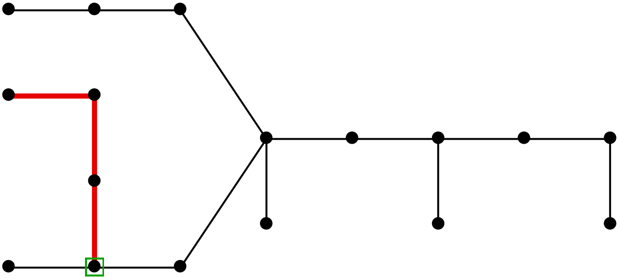
The Extremal Graphs

Thm. (LeSaulnier–West [2013]) An n-vertex graph G satisfies $a_t(G) = \frac{n+1}{3}$ if and only if G is a tree obtained from P_2 by iteratively growing a 3-edge path from a neighbor of a leaf.



The Extremal Graphs

Thm. (LeSaulnier–West [2013]) An n-vertex graph G satisfies $a_t(G) = \frac{n+1}{3}$ if and only if G is a tree obtained from P_2 by iteratively growing a 3-edge path from a neighbor of a leaf.



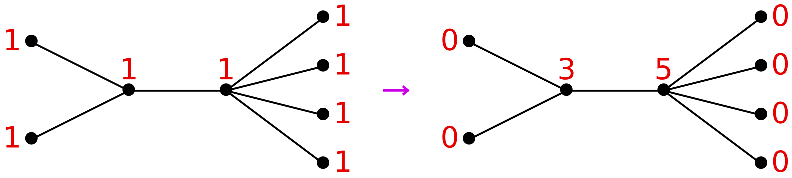
• The graphs G such that $a_u(G) = \frac{n+1}{3}$ are precisely those such that $a_t(G) = \frac{n+1}{3}$.

Bounds for n-vertex Trees (LPWWW [2013])

- For diameter 6 or higher, $\max a_t(T) = \left\lfloor \frac{n+1}{3} \right\rfloor$.

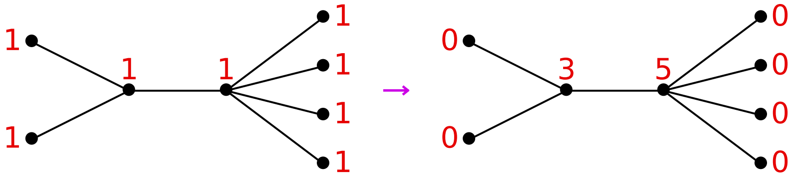
Bounds for n-vertex Trees (LPWWW [2013])

- For diameter 6 or higher, $\max a_t(T) = \left\lfloor \frac{n+1}{3} \right\rfloor$.
- For diameter 2 or 3, $a_t(T) = 1$. "double-stars"



Bounds for n-vertex Trees (LPWWW [2013])

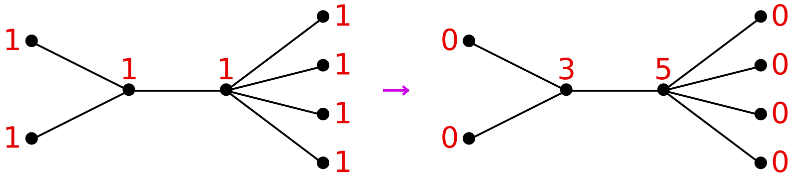
- For diameter 6 or higher, $\max a_t(T) = \left\lfloor \frac{n+1}{3} \right\rfloor$.
- For diameter 2 or 3, $a_t(T) = 1$. "double-stars"



What about diameter 4 and diameter 5?

Bounds for n-vertex Trees (LPWWW [2013])

- For diameter 6 or higher, $\max a_t(T) = \left\lfloor \frac{n+1}{3} \right\rfloor$.
- For diameter 2 or 3, $a_t(T) = 1$. "double-stars"

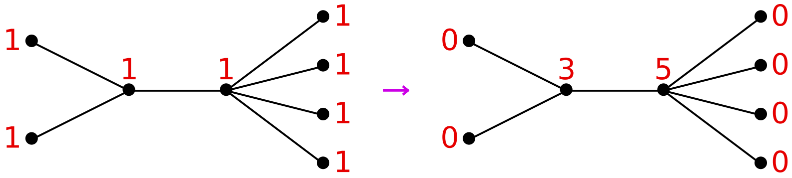


What about diameter 4 and diameter 5?

Thm. For diameter 4 or 5, $\max a_t(T) = \Theta(\sqrt{n \lg n})$.

Bounds for n-vertex Trees (LPWWW [2013])

- For diameter 6 or higher, $\max a_t(T) = \left\lfloor \frac{n+1}{3} \right\rfloor$.
- For diameter 2 or 3, $a_t(T) = 1$. "double-stars"

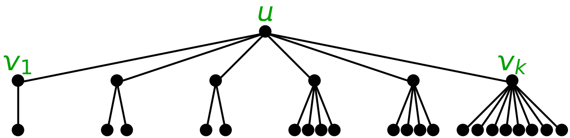


What about diameter 4 and diameter 5?

Thm. For diameter 4 or 5, $\max a_t(T) = \Theta(\sqrt{n \lg n})$.

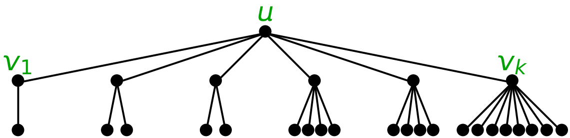
- When $\mathrm{diam}(T) = 5$, delete the central edge and use the result for trees of diameter 4.

Trees with Diameter 4, Upper Bound



Thm. $a_t(T) \leq \sqrt{n \lg n}$.

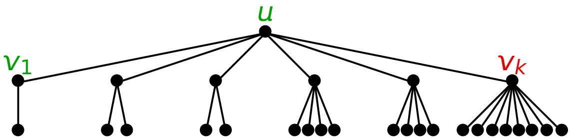
Trees with Diameter 4, Upper Bound



Thm. $a_t(T) \le \sqrt{n \lg n}$.

Pf. Case 1: $k \le \sqrt{n \lg n} \Rightarrow N(u)$ absorbs all.

Trees with Diameter 4, Upper Bound

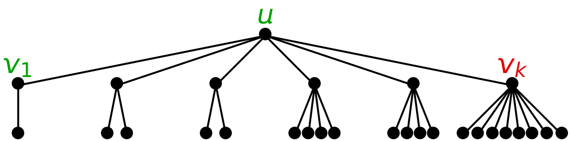


Thm. $a_t(T) \le \sqrt{n \lg n}$.

Pf. Case 1: $k \le \sqrt{n \lg n} \Rightarrow N(u)$ absorbs all.

Case 2: $d(v_k) \ge \sqrt{n} \Rightarrow a_t(T) \le 1 + \sqrt{(n - \sqrt{n}) \lg n}$.

Trees with Diameter 4, Upper Bound



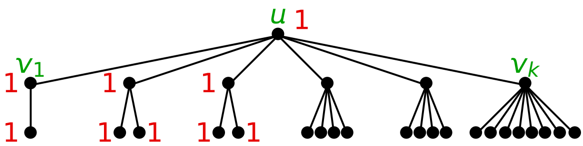
Thm. $a_t(T) \le \sqrt{n \lg n}$.

Pf. Case 1: $k \le \sqrt{n \lg n} \Rightarrow N(u)$ absorbs all.

Case 2: $d(v_k) \ge \sqrt{n} \Rightarrow a_t(T) \le 1 + \sqrt{(n - \sqrt{n}) \lg n}$.

Case 3: let w_i = weight on u before processing v_i;
algorithm moves weight $\min\{w_i, d(v_i)\}$ from v_i to u.

Trees with Diameter 4, Upper Bound



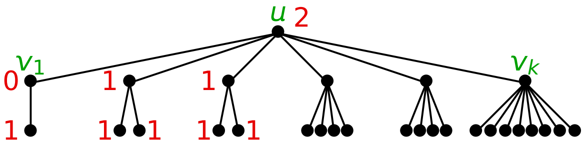
Thm. $a_t(T) \le \sqrt{n \lg n}$.

Pf. Case 1: $k \le \sqrt{n \lg n} \Rightarrow N(u)$ absorbs all.

Case 2: $d(v_k) \ge \sqrt{n} \Rightarrow a_t(T) \le 1 + \sqrt{(n - \sqrt{n}) \lg n}$.

Case 3: let w_i = weight on u before processing v_i;
algorithm moves weight $\min\{w_i, d(v_i)\}$ from v_i to u.

Trees with Diameter 4, Upper Bound



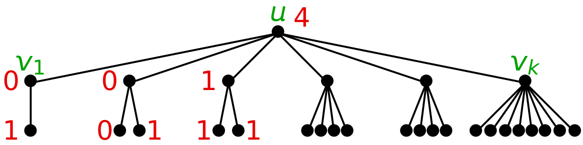
Thm. $a_t(T) \le \sqrt{n \lg n}$.

Pf. Case 1: $k \le \sqrt{n \lg n} \Rightarrow N(u)$ absorbs all.

Case 2: $d(v_k) \ge \sqrt{n} \Rightarrow a_t(T) \le 1 + \sqrt{(n - \sqrt{n}) \lg n}$.

Case 3: let w_i = weight on u before processing v_i;
algorithm moves weight $\min\{w_i, d(v_i)\}$ from v_i to u.

Trees with Diameter 4, Upper Bound



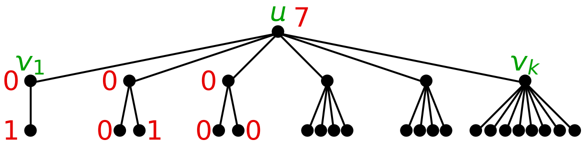
Thm. $a_t(T) \leq \sqrt{n \lg n}$.

Pf. Case 1: $k \leq \sqrt{n \lg n} \Rightarrow N(u)$ absorbs all.

Case 2: $d(v_k) \geq \sqrt{n} \Rightarrow a_t(T) \leq 1 + \sqrt{(n - \sqrt{n}) \lg n}$.

Case 3: let w_i = weight on u before processing v_i; algorithm moves weight $\min\{w_i, d(v_i)\}$ from v_i to u.

Trees with Diameter 4, Upper Bound



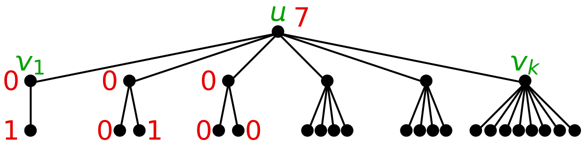
Thm. $a_t(T) \le \sqrt{n \lg n}$.

Pf. Case 1: $k \le \sqrt{n \lg n} \Rightarrow N(u)$ absorbs all.

Case 2: $d(v_k) \ge \sqrt{n} \Rightarrow a_t(T) \le 1 + \sqrt{(n - \sqrt{n}) \lg n}$.

Case 3: let w_i = weight on u before processing v_i;
algorithm moves weight $\min\{w_i, d(v_i)\}$ from v_i to u.

Trees with Diameter 4, Upper Bound



Thm. $a_t(T) \le \sqrt{n \lg n}$.

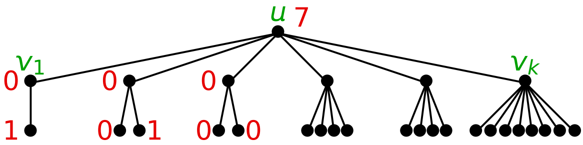
Pf. Case 1: $k \le \sqrt{n \lg n} \Rightarrow N(u)$ absorbs all.

Case 2: $d(v_k) \ge \sqrt{n} \Rightarrow a_t(T) \le 1 + \sqrt{(n - \sqrt{n}) \lg n}$.

Case 3: let w_i = weight on u before processing v_i;
algorithm moves weight $\min\{w_i, d(v_i)\}$ from v_i to u.

Let $S = \{i : d(v_i) > w_i\}$; some of $N(v_i)$ stays when $i \in S$.

Trees with Diameter 4, Upper Bound



Thm. $a_t(T) \le \sqrt{n \lg n}$.

Pf. Case 1: $k \le \sqrt{n \lg n}$ $\Rightarrow$ $N(u)$ absorbs all.

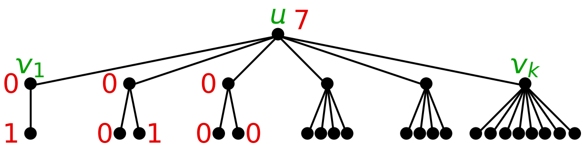
Case 2: $d(v_k) \ge \sqrt{n}$ $\Rightarrow$ $a_t(T) \le 1 + \sqrt{(n - \sqrt{n}) \lg n}$.

Case 3: let w_i = weight on u before processing v_i;
algorithm moves weight $\min\{w_i, d(v_i)\}$ from v_i to u.

Let $S = \{i : d(v_i) > w_i\}$; some of $N(v_i)$ stays when $i \in S$.

Let $m = \max S$. Weight doubles $\Rightarrow$ $|S| \le \lg d(v_m)$.

Trees with Diameter 4, Upper Bound



Thm. $a_t(T) \le \sqrt{n \lg n}$.

Pf. Case 1: $k \le \sqrt{n \lg n} \Rightarrow N(u)$ absorbs all.

Case 2: $d(v_k) \ge \sqrt{n} \Rightarrow a_t(T) \le 1 + \sqrt{(n - \sqrt{n}) \lg n}$.

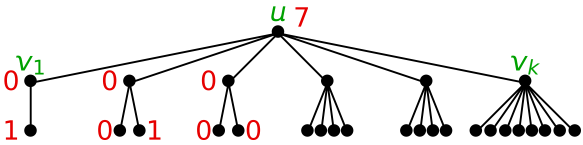
Case 3: let w_i = weight on u before processing v_i; algorithm moves weight $\min\{w_i, d(v_i)\}$ from v_i to u.

Let $S = \{i : d(v_i) > w_i\}$; some of $N(v_i)$ stays when $i \in S$.

Let $m = \max S$. Weight doubles $\Rightarrow |S| \le \lg d(v_m)$.

$a_t(T) \le d(v_m) \lg d(v_m)$ (later weight goes to u).

Trees with Diameter 4, Upper Bound



Thm. $a_t(T) \le \sqrt{n \lg n}$.

Pf. Case 1: $k \le \sqrt{n \lg n} \Rightarrow N(u)$ absorbs all.

Case 2: $d(v_k) \ge \sqrt{n} \Rightarrow a_t(T) \le 1 + \sqrt{(n - \sqrt{n}) \lg n}$.

Case 3: let w_i = weight on u before processing v_i; algorithm moves weight $\min\{w_i, d(v_i)\}$ from v_i to u.

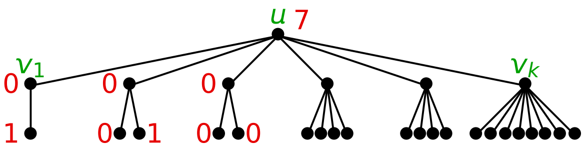
Let $S = \{i : d(v_i) > w_i\}$; some of $N(v_i)$ stays when $i \in S$.

Let $m = \max S$. Weight doubles $\Rightarrow |S| \le \lg d(v_m)$.

$a_t(T) \le d(v_m) \lg d(v_m)$ (later weight goes to u).

$$m \le w_m < d(v_m) \le d(v_k) < \sqrt{n} < k/2.$$

Trees with Diameter 4, Upper Bound



Thm. $a_t(T) \le \sqrt{n \lg n}$.

Pf. Case 1: $k \le \sqrt{n \lg n} \Rightarrow N(u)$ absorbs all.

Case 2: $d(v_k) \ge \sqrt{n} \Rightarrow a_t(T) \le 1 + \sqrt{(n - \sqrt{n}) \lg n}$.

Case 3: let w_i = weight on u before processing v_i;
algorithm moves weight $\min\{w_i, d(v_i)\}$ from v_i to u.

Let $S = \{i : d(v_i) > w_i\}$; some of $N(v_i)$ stays when $i \in S$.

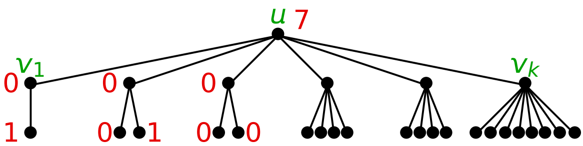
Let $m = \max S$. Weight doubles $\Rightarrow |S| \le \lg d(v_m)$.

$a_t(T) \le d(v_m) \lg d(v_m)$ (later weight goes to u).

$$m \le w_m < d(v_m) \le d(v_k) < \sqrt{n} < k/2.$$

$d(v_m) < \frac{n-m}{k-m} < \frac{2n}{k}$ (using $m < k/2$).

Trees with Diameter 4, Upper Bound



Thm. $a_t(T) \le \sqrt{n \lg n}$.

Pf. Case 1: $k \le \sqrt{n \lg n} \Rightarrow N(u)$ absorbs all.

Case 2: $d(v_k) \ge \sqrt{n} \Rightarrow a_t(T) \le 1 + \sqrt{(n - \sqrt{n}) \lg n}$.

Case 3: let w_i = weight on u before processing v_i; algorithm moves weight $\min\{w_i, d(v_i)\}$ from v_i to u.

Let $S = \{i : d(v_i) > w_i\}$; some of $N(v_i)$ stays when $i \in S$.

Let $m = \max S$. Weight doubles $\Rightarrow |S| \le \lg d(v_m)$.

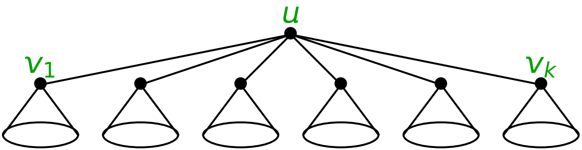
$a_t(T) \le d(v_m) \lg d(v_m)$ (later weight goes to u).

$$m \le w_m < d(v_m) \le d(v_k) < \sqrt{n} < k/2.$$

$d(v_m) < \frac{n-m}{k-m} < \frac{2n}{k}$ (using $m < k/2$).

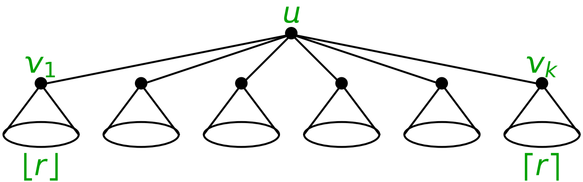
Hence $a_t(T) < \frac{2n}{k} \lg \frac{2n}{k} < \sqrt{n \lg n}$. ∎

Trees with Diameter 4, Lower Bound



Thm. $a_t(T) \geq (1 - o(1))\sqrt{.5n \lg n}$.

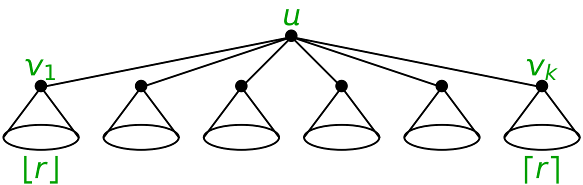
Trees with Diameter 4, Lower Bound



Thm. $a_t(T) \ge (1 - o(1))\sqrt{.5n \lg n}$.

Pf. Let $r = \sqrt{2n/\lg n}$ and $k = \left\lfloor \frac{n-1}{r+1} \right\rfloor \approx \sqrt{.5n \lg n}$.

Trees with Diameter 4, Lower Bound

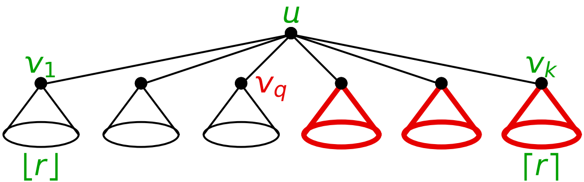


Thm. $a_t(T) \geq (1 - o(1))\sqrt{.5 n \lg n}$.

Pf. Let $r = \sqrt{2n/\lg n}$ and $k = \left\lfloor \frac{n-1}{r+1} \right\rfloor \approx \sqrt{.5 n \lg n}$.

Let q = #nbrs giving weight to u in optimal algorithm.
We may assume they are $v_1, \ldots, v_q$ in order.

Trees with Diameter 4, Lower Bound



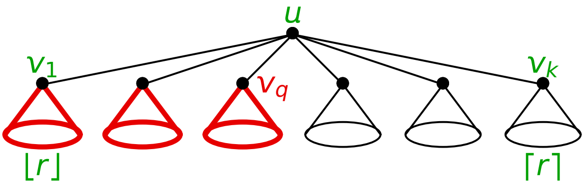
Thm. $a_t(T) \geq (1 - o(1))\sqrt{.5n \lg n}$.

Pf. Let $r = \sqrt{2n/\lg n}$ and $k = \left\lfloor \frac{n-1}{r+1} \right\rfloor \approx \sqrt{.5n \lg n}$.

Let $q =$ #nbrs giving weight to u in optimal algorithm.
We may assume they are $v_1, \ldots, v_q$ in order.

If $q < \lg r$, then weight remains in $k - o(k)$ subtrees.

Trees with Diameter 4, Lower Bound



Thm. $a_t(T) \geq (1 - o(1))\sqrt{.5 n \lg n}$.

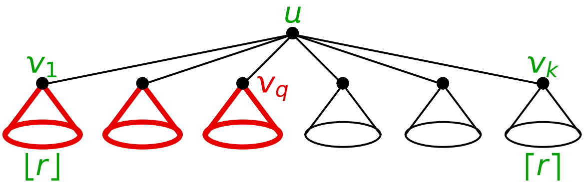
Pf. Let $r = \sqrt{2n/\lg n}$ and $k = \left\lfloor \frac{n-1}{r+1} \right\rfloor \approx \sqrt{.5 n \lg n}$.

Let $q = \#$nbrs giving weight to u in optimal algorithm.
We may assume they are $v_1, \ldots, v_q$ in order.

If $q < \lg r$, then weight remains in $k - o(k)$ subtrees.

If $q \geq \lg r$, then at least $r - (2^{i-1} - 1)$ leaf neighbors of v_i are stranded, for $i \leq \lg r$.

Trees with Diameter 4, Lower Bound



Thm. $a_t(T) \geq (1 - o(1))\sqrt{.5n \lg n}$.

Pf. Let $r = \sqrt{2n/\lg n}$ and $k = \left\lfloor \frac{n-1}{r+1} \right\rfloor \approx \sqrt{.5n \lg n}$.

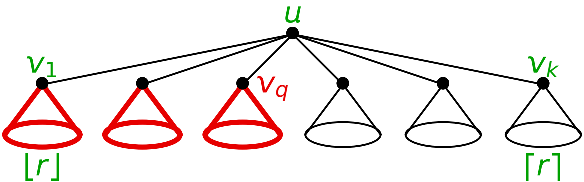
Let $q = \#$nbrs giving weight to u in optimal algorithm.
We may assume they are $v_1, \ldots, v_q$ in order.

If $q < \lg r$, then weight remains in $k - o(k)$ subtrees.

If $q \geq \lg r$, then at least $r - (2^{i-1} - 1)$ leaf neighbors of v_i are stranded, for $i \leq \lg r$.

Thus #leaves stranded $\geq r \lfloor \lg r \rfloor - \sum_{i=1}^{\lfloor \lg r \rfloor} (2^{i-1} - 1)$

Trees with Diameter 4, Lower Bound



Thm. $a_t(T) \geq (1 - o(1))\sqrt{.5 n \lg n}$.

Pf. Let $r = \sqrt{2n/\lg n}$ and $k = \left\lfloor \frac{n-1}{r+1} \right\rfloor \approx \sqrt{.5 n \lg n}$.

Let $q = \#$nbrs giving weight to u in optimal algorithm.
We may assume they are $v_1, \ldots, v_q$ in order.

If $q < \lg r$, then weight remains in $k - o(k)$ subtrees.

If $q \geq \lg r$, then at least $r - (2^{i-1} - 1)$ leaf neighbors of v_i are stranded, for $i \leq \lg r$.

Thus #leaves stranded $\geq r \lfloor \lg r \rfloor - \sum_{i=1}^{\lfloor \lg r \rfloor} (2^{i-1} - 1)$

$= (1 - o(1))(r \lg r) = (1 - o(1))\sqrt{.5 n \lg n}$. ■

Complexity and $a_t = 1$

Trees with $a_t(T) = 1$ are recognizable in quadratic time (Cai [1993]).

Complexity and $a_t = 1$

Trees with $a_t(T) = 1$ are recognizable in quadratic time (Cai [1993]).

Thm. On trees, $a_t(T) \le k$ is testable in time $O(n^{k+1})$.

Pf. Try all sets of $k-1$ edges to delete and form k components. Test each for being a union tree. ∎

Complexity and $a_t = 1$

Trees with $a_t(T) = 1$ are recognizable in quadratic time (Cai [1993]).

Thm. On trees, $a_t(T) \le k$ is testable in time $O(n^{k+1})$.

Pf. Try all sets of $k-1$ edges to delete and form k components. Test each for being a union tree. ■

Thm. (Lampert–Slater [1995]) Testing $a_t(G) = 1$ on general graphs is NP-complete.

Complexity and $a_t = 1$

Trees with $a_t(T) = 1$ are recognizable in quadratic time (Cai [1993]).

Thm. On trees, $a_t(T) \le k$ is testable in time $O(n^{k+1})$.

Pf. Try all sets of $k-1$ edges to delete and form k components. Test each for being a union tree. ∎

Thm. (Lampert–Slater [1995]) Testing $a_t(G) = 1$ on general graphs is NP-complete.

Sufficient Conditions for $a_t(G) = 1$:

Complexity and $\alpha_t = 1$

Trees with $\alpha_t(T) = 1$ are recognizable in quadratic time (Cai [1993]).

Thm. On trees, $\alpha_t(T) \le k$ is testable in time $O(n^{k+1})$.

Pf. Try all sets of $k-1$ edges to delete and form k components. Test each for being a union tree. ∎

Thm. (Lampert–Slater [1995]) Testing $\alpha_t(G) = 1$ on general graphs is NP-complete.

Sufficient Conditions for $\alpha_t(G) = 1$:

- $\exists v$ with $d(v) \ge \frac{n}{2}$ such that $N(v)$ is a dominating set.

Complexity and $a_t = 1$

Trees with $a_t(T) = 1$ are recognizable in quadratic time (Cai [1993]).

Thm. On trees, $a_t(T) \leq k$ is testable in time $O(n^{k+1})$.

Pf. Try all sets of $k-1$ edges to delete and form k components. Test each for being a union tree. ∎

Thm. (Lampert–Slater [1995]) Testing $a_t(G) = 1$ on general graphs is NP-complete.

Sufficient Conditions for $a_t(G) = 1$:

- $\exists v$ with $d(v) \geq \frac{n}{2}$ such that $N(v)$ is a dominating set.
- G is $(n-1)/2$-regular.

Complexity and $a_t = 1$

Trees with $a_t(T) = 1$ are recognizable in quadratic time (Cai [1993]).

Thm. On trees, $a_t(T) \le k$ is testable in time $O(n^{k+1})$.

Pf. Try all sets of $k-1$ edges to delete and form k components. Test each for being a union tree. ■

Thm. (Lampert–Slater [1995]) Testing $a_t(G) = 1$ on general graphs is NP-complete.

Sufficient Conditions for $a_t(G) = 1$:

- $\exists v$ with $d(v) \ge \frac{n}{2}$ such that $N(v)$ is a dominating set.
- G is $(n-1)/2$-regular.

Thm. If $G \ne C_5$, then $a_t(G) = 1$ or $a_t(\overline{G}) = 1$.

Complexity and $a_t = 1$

Trees with $a_t(T) = 1$ are recognizable in quadratic time (Cai [1993]).

Thm. On trees, $a_t(T) \le k$ is testable in time $O(n^{k+1})$.

Pf. Try all sets of $k-1$ edges to delete and form k components. Test each for being a union tree. ■

Thm. (Lampert–Slater [1995]) Testing $a_t(G) = 1$ on general graphs is NP-complete.

Sufficient Conditions for $a_t(G) = 1$:

- $\exists v$ with $d(v) \ge \frac{n}{2}$ such that $N(v)$ is a dominating set.
- G is $(n-1)/2$-regular.

Thm. If $G \ne C_5$, then $a_t(G) = 1$ or $a_t(\overline{G}) = 1$.

Thm. If $G \ne C_5$ and $d(u) + d(v) \ge n - 1$ whenever $uv \notin E(G)$, then $a_t(G) = 1$.

Other Results on Total Acquisition

Other Results on Total Acquisition

Edge-deletion

Thm. If $e \in E(G)$, then $a_t(G - e) < a_t(G) + 7\sqrt{n}$.

Other Results on Total Acquisition

Edge-deletion

Thm. If $e \in E(G)$, then $a_t(G - e) < a_t(G) + 7\sqrt{n}$.

Thm. There exists a tree T with an edge e such that $a_t(T) = 1$ and $a_t(T - e) > \sqrt{n}/2$.

Other Results on Total Acquisition

Edge-deletion

Thm. If $e \in E(G)$, then $a_t(G - e) < a_t(G) + 7\sqrt{n}$.

Thm. There exists a tree T with an edge e such that $a_t(T) = 1$ and $a_t(T - e) > \sqrt{n}/2$.

Diameter 2

Thm. $\mathrm{diam} G = 2 \Rightarrow a_t(G) \le 250 \lg n \lg \lg n$.

Other Results on Total Acquisition

Edge-deletion

Thm. If $e \in E(G)$, then $a_t(G - e) < a_t(G) + 7\sqrt{n}$.

Thm. There exists a tree T with an edge e such that $a_t(T) = 1$ and $a_t(T - e) > \sqrt{n}/2$.

Diameter 2

Thm. $\text{diam}\, G = 2 \quad \Rightarrow \quad a_t(G) \le 250 \lg n \lg\lg n$.

Thm. $\text{diam}\, G = 2$ & $C_4 \not\subseteq G$ & $\Delta(G) \ge 8 \quad \Rightarrow \quad a_t(G) = 1$.

Other Results on Total Acquisition

Edge-deletion

Thm. If $e \in E(G)$, then $a_t(G - e) < a_t(G) + 7\sqrt{n}$.

Thm. There exists a tree T with an edge e such that $a_t(T) = 1$ and $a_t(T - e) > \sqrt{n}/2$.

Diameter 2

Thm. $\operatorname{diam} G = 2 \quad \Rightarrow \quad a_t(G) \leq 250 \lg n \lg \lg n$.

Thm. $\operatorname{diam} G = 2$ & $C_4 \not\subseteq G$ & $\Delta(G) \geq 8 \quad \Rightarrow \quad a_t(G) = 1$.

Conj. $\exists\, c$ such that $\operatorname{diam} G = 2 \quad \Rightarrow \quad a_t(G) \leq c$.

Other Results on Total Acquisition

Edge-deletion

Thm. If $e \in E(G)$, then $a_t(G-e) < a_t(G) + 7\sqrt{n}$.

Thm. There exists a tree T with an edge e such that $a_t(T) = 1$ and $a_t(T-e) > \sqrt{n}/2$.

Diameter 2

Thm. $\mathrm{diam} G = 2 \Rightarrow a_t(G) \le 250 \lg n \lg\lg n$.

Thm. $\mathrm{diam} G = 2$ & $C_4 \not\subseteq G$ & $\Delta(G) \ge 8 \Rightarrow a_t(G) = 1$.

Conj. $\exists\, c$ such that $\mathrm{diam} G = 2 \Rightarrow a_t(G) \le c$.

Perhaps $c = 2$. This suffices for Moore graphs, polarity graphs, and graphs without 4-cycles.

Other Open Problems (for Total Acquisition)

Conj. For random graphs,

Other Open Problems (for Total Acquisition)

Conj. For random graphs,

$\exists\ c$ such that $p_n \geq \sqrt{\frac{c \ln n}{n}} \Rightarrow$ almost always $a_t(G) = 1$.

Other Open Problems (for Total Acquisition)

Conj. For random graphs,

$\exists\, c$ such that $p_n \geq \sqrt{\frac{c \ln n}{n}} \Rightarrow$ almost always $a_t(G) = 1$.

$\exists\, c$ such that $p_n \leq c/n \Rightarrow$ almost always $a_t(G) = \Theta(n)$.

Conj. For almost all trees, $a_t(T) = \Theta(n)$.

Other Open Problems (for Total Acquisition)

Conj. For random graphs,

$\exists\, c$ such that $p_n \ge \sqrt{\frac{c \ln n}{n}} \Rightarrow$ almost always $a_t(G) = 1$.

$\exists\, c$ such that $p_n \le c/n \Rightarrow$ almost always $a_t(G) = \Theta(n)$.

Conj. For almost all trees, $a_t(T) = \Theta(n)$.

Ques. What is the maximum of $a_t(G)$ when G is a connected n-vertex graph with minimum degree k?

Other Open Problems (for Total Acquisition)

Conj. For random graphs,

$\exists\, c$ such that $p_n \geq \sqrt{\frac{c \ln n}{n}} \Rightarrow$ almost always $a_t(G) = 1$.

$\exists\, c$ such that $p_n \leq c/n \Rightarrow$ almost always $a_t(G) = \Theta(n)$.

Conj. For almost all trees, $a_t(T) = \Theta(n)$.

Ques. What is the maximum of $a_t(G)$ when G is a connected n-vertex graph with minimum degree k?

Always $a_t(G) \leq \frac{1+\ln(k+1)}{k+1} n$, since $a_t(G) \leq \gamma(G)$.

Other Open Problems (for Total Acquisition)

Conj. For random graphs,

$\exists\, c$ such that $p_n \geq \sqrt{\frac{c \ln n}{n}} \;\Rightarrow\;$ almost always $a_t(G) = 1$.

$\exists\, c$ such that $p_n \leq c/n \;\Rightarrow\;$ almost always $a_t(G) = \Theta(n)$.

Conj. For almost all trees, $a_t(T) = \Theta(n)$.

Ques. What is the maximum of $a_t(G)$ when G is a connected n-vertex graph with minimum degree k?

Always $a_t(G) \leq \frac{1+\ln(k+1)}{k+1} n$, since $a_t(G) \leq \gamma(G)$.

For $k = 2$, we know $a_t(C_n) = \lceil n/4 \rceil$, but another construction has a larger value.

Other Open Problems (for Total Acquisition)

Conj. For random graphs,

$\exists\, c$ such that $p_n \geq \sqrt{\frac{c \ln n}{n}} \;\Rightarrow\;$ almost always $a_t(G) = 1$.

$\exists\, c$ such that $p_n \leq c/n \;\Rightarrow\;$ almost always $a_t(G) = \Theta(n)$.

Conj. For almost all trees, $a_t(T) = \Theta(n)$.

Ques. What is the maximum of $a_t(G)$ when G is a connected n-vertex graph with minimum degree k?

Always $a_t(G) \leq \frac{1+\ln(k+1)}{k+1} n$, since $a_t(G) \leq \gamma(G)$.

For $k = 2$, we know $a_t(C_n) = \lceil n/4 \rceil$, but another construction has a larger value.

For a binary tree with triangles appended at the leaves, $\delta(G) = 2$ but $a_t(G) > (\frac{1}{4} + \frac{1}{1024})n$.

Unit Acquisition — Always $a_u(G) \leq a_t(G)$

Def. An ascending tree is a rooted tree such that each leaf has weight at most that of its neighbor, and other weights strictly increase along paths to the root.

Unit Acquisition — Always $a_u(G) \leq a_t(G)$

Def. An ascending tree is a rooted tree such that each leaf has weight at most that of its neighbor, and other weights strictly increase along paths to the root.

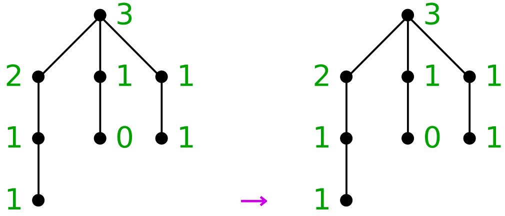
Lem. All weight in an ascending tree can be acquired to the root by unit acquisition moves.

Unit Acquisition — Always $a_u(G) \leq a_t(G)$

Def. An ascending tree is a rooted tree such that each leaf has weight at most that of its neighbor, and other weights strictly increase along paths to the root.

Lem. All weight in an ascending tree can be acquired to the root by unit acquisition moves.

Pf. One unit can be moved from any leaf to the root. ■

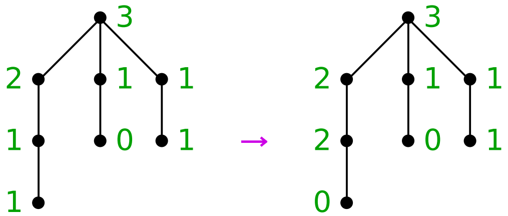


Unit Acquisition — Always $\alpha_u(G) \leq \alpha_t(G)$

Def. An ascending tree is a rooted tree such that each leaf has weight at most that of its neighbor, and other weights strictly increase along paths to the root.

Lem. All weight in an ascending tree can be acquired to the root by unit acquisition moves.

Pf. One unit can be moved from any leaf to the root. ■

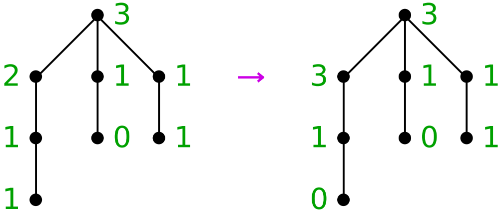


Unit Acquisition — Always $\alpha_u(G) \le \alpha_t(G)$

Def. An ascending tree is a rooted tree such that each leaf has weight at most that of its neighbor, and other weights strictly increase along paths to the root.

Lem. All weight in an ascending tree can be acquired to the root by unit acquisition moves.

Pf. One unit can be moved from any leaf to the root. ■

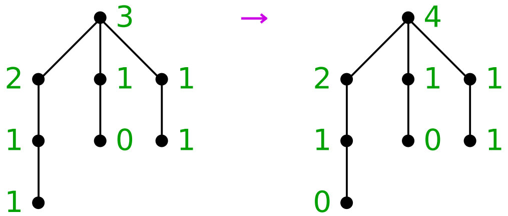


Unit Acquisition — Always $\alpha_u(G) \le \alpha_t(G)$

Def. An ascending tree is a rooted tree such that each leaf has weight at most that of its neighbor, and other weights strictly increase along paths to the root.

Lem. All weight in an ascending tree can be acquired to the root by unit acquisition moves.

Pf. One unit can be moved from any leaf to the root. ■

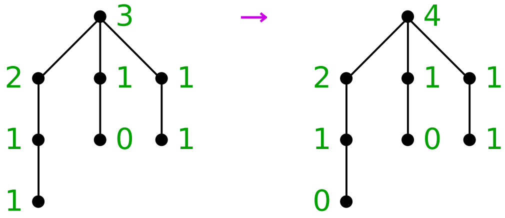


Unit Acquisition — Always $a_u(G) \leq a_t(G)$

Def. An ascending tree is a rooted tree such that each leaf has weight at most that of its neighbor, and other weights strictly increase along paths to the root.

Lem. All weight in an ascending tree can be acquired to the root by unit acquisition moves.

Pf. One unit can be moved from any leaf to the root. ∎



Thm. $a_u(T) = 1 \iff$ an ascending tree can be produced.

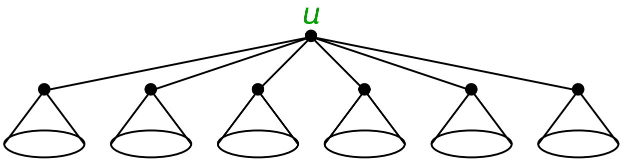
Trees of Diameter 4

- $\max \alpha_t(T) = \Theta(\sqrt{n \lg n})$.

Trees of Diameter 4

- $\max a_t(T) = \Theta(\sqrt{n \lg n})$.

Thm. For trees of diameter 4, $\max a_u(T) = \left\lfloor \sqrt{n-1} \right\rfloor$.

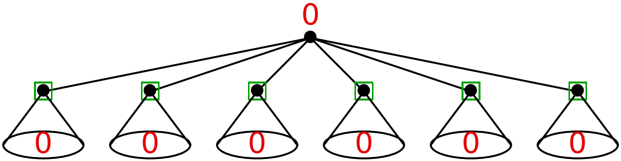


Trees of Diameter 4

- $\max a_t(T) = \Theta(\sqrt{n \lg n})$.

Thm. For trees of diameter 4, $\max a_u(T) = \lfloor \sqrt{n-1} \rfloor$.

Pf. **Upper Bd:** If $d(u) \le \sqrt{n-1}$, then $N(u)$ absorbs all.



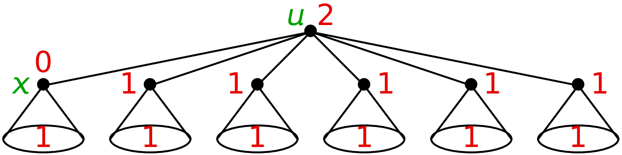
Trees of Diameter 4

- $\max a_t(T) = \Theta(\sqrt{n \lg n})$.

Thm. For trees of diameter 4, $\max a_u(T) = \lfloor \sqrt{n-1} \rfloor$.

Pf. **Upper Bd:** If $d(u) \le \sqrt{n-1}$, then $N(u)$ absorbs all.

Else $\exists\, x \in N(u)$ with $d(x) < \sqrt{n-1}$. Moving $x \to u$ makes an ascending tree omitting $< \lfloor \sqrt{n-1} \rfloor$ chips.



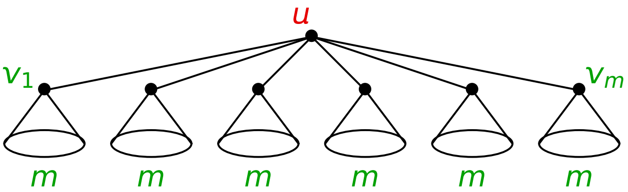
Trees of Diameter 4

- $\max a_t(T) = \Theta(\sqrt{n \lg n})$.

Thm. For trees of diameter 4, $\max a_u(T) = \lfloor \sqrt{n-1} \rfloor$.

Pf. Upper Bd: If $d(u) \le \sqrt{n-1}$, then $N(u)$ absorbs all.

Else $\exists\, x \in N(u)$ with $d(x) < \sqrt{n-1}$. Moving $x \to u$ makes an ascending tree omitting $< \lfloor \sqrt{n-1} \rfloor$ chips.



Lower Bound: Make tree with $d(u) = m$ and $d(v) = m$ for $v \in N(u)$, so $n = m^2 + 1$. The first move involving u makes at least m components with positive weight.

Degree Bounds

- If $\delta(G) = k$, then $\alpha_t(G) \leq \gamma(G) \leq \frac{1+\ln(k+1)}{k+1} n$.

Degree Bounds

- If $\delta(G) = k$, then $\alpha_t(G) \le \gamma(G) \le \frac{1+\ln(k+1)}{k+1} n$.

Thm. If $\delta(G) = k$, then $\alpha_u(G) \le \frac{1}{k} n$.

Degree Bounds

- If $\delta(G) = k$, then $a_t(G) \leq \gamma(G) \leq \frac{1+\ln(k+1)}{k+1} n$.

Thm. If $\delta(G) = k$, then $a_u(G) \leq \frac{1}{k} n$.

Pf. **Idea:** Partition $V(G)$ into trees of diameter 4; acquire to $1/k$ of the vertices in each tree. ■

Degree Bounds

- If $\delta(G) = k$, then $\alpha_t(G) \le \gamma(G) \le \frac{1+\ln(k+1)}{k+1} n$.

Thm. If $\delta(G) = k$, then $\alpha_u(G) \le \frac{1}{k} n$.

Pf. Idea: Partition $V(G)$ into trees of diameter 4; acquire to $1/k$ of the vertices in each tree. ■

- (Lampert–Slater [1995]) $\alpha_t(G) \ge n/2^{\Delta(G)}$, since the weight of vertex v can never exceed $2^{d(v)}$.

Degree Bounds

- If $\delta(G) = k$, then $a_t(G) \le \gamma(G) \le \frac{1+\ln(k+1)}{k+1} n$.

Thm. If $\delta(G) = k$, then $a_u(G) \le \frac{1}{k} n$.

Pf. Idea: Partition $V(G)$ into trees of diameter 4; acquire to $1/k$ of the vertices in each tree. ■

- (Lampert–Slater [1995]) $a_t(G) \ge n/2^{\Delta(G)}$, since the weight of vertex v can never exceed $2^{d(v)}$.

- Max vertex weight reachable under unit acquisition:

$\Delta(G) \rightarrow$	1	2	3	4	5
max wt $\rightarrow$	1	4	10	239	???

Degree Bounds

- If $\delta(G) = k$, then $a_t(G) \le \gamma(G) \le \frac{1+\ln(k+1)}{k+1} n$.

Thm. If $\delta(G) = k$, then $a_u(G) \le \frac{1}{k} n$.

Pf. Idea: Partition $V(G)$ into trees of diameter 4; acquire to $1/k$ of the vertices in each tree. ∎

- (Lampert–Slater [1995]) $a_t(G) \ge n/2^{\Delta(G)}$, since the weight of vertex v can never exceed $2^{d(v)}$.

- Max vertex weight reachable under unit acquisition:

$\Delta(G) \rightarrow$	1	2	3	4	5
max wt $\rightarrow$	1	4	10	239	???

Thm. (Wenger) There is an infinite family of trees with maximum degree 5 and unit acquisition number 1.

Diameter 2

- **Conj:** $\alpha_t(G) \leq 2$ when G has diameter 2.

Diameter 2

- **Conj:** $\alpha_t(G) \le 2$ when G has diameter 2.

Thm. (Wenger) If $\operatorname{diam} G = 2$ and G is not C_5 or the Petersen graph, then $\alpha_u(G) = 1$.

Diameter 2

- **Conj:** $\alpha_t(G) \le 2$ when G has diameter 2.

Thm. (Wenger) If $\operatorname{diam} G = 2$ and G is not C_5 or the Petersen graph, then $\alpha_u(G) = 1$.

Pf. **(Idea)** In each of several cases depending on neighborhoods within a largest clique, a few moves create an ascending tree on the remaining vertices. ∎

Diameter 2

- **Conj:** $\alpha_t(G) \le 2$ when G has diameter 2.

Thm. (Wenger) If $\operatorname{diam} G = 2$ and G is not C_5 or the Petersen graph, then $\alpha_u(G) = 1$.

Pf. **(Idea)** In each of several cases depending on neighborhoods within a largest clique, a few moves create an ascending tree on the remaining vertices. ∎

Open Problems:

Diameter 2

- **Conj:** $a_t(G) \leq 2$ when G has diameter 2.

Thm. (Wenger) If $\operatorname{diam} G = 2$ and G is not C_5 or the Petersen graph, then $a_u(G) = 1$.

Pf. **(Idea)** In each of several cases depending on neighborhoods within a largest clique, a few moves create an ascending tree on the remaining vertices. ■

Open Problems:

- Find $\max(a_u(G))$ when $|V(G)| = n$ and $\delta(G) = k$.

Diameter 2

- **Conj:** $a_t(G) \le 2$ when G has diameter 2.

Thm. (Wenger) If $\operatorname{diam} G = 2$ and G is not C_5 or the Petersen graph, then $a_u(G) = 1$.

Pf. **(Idea)** In each of several cases depending on neighborhoods within a largest clique, a few moves create an ascending tree on the remaining vertices. ■

Open Problems:

- Find $\max(a_u(G))$ when $|V(G)| = n$ and $\delta(G) = k$.

- Characterize the trees with $a_u(T) = 1$.

(We think we know which caterpillars work.)

Diameter 2

- **Conj:** $a_t(G) \leq 2$ when G has diameter 2.

Thm. (Wenger) If $\text{diam}\, G = 2$ and G is not C_5 or the Petersen graph, then $a_u(G) = 1$.

Pf. **(Idea)** In each of several cases depending on neighborhoods within a largest clique, a few moves create an ascending tree on the remaining vertices. ■

Open Problems:

- Find $\max(a_u(G))$ when $|V(G)| = n$ and $\delta(G) = k$.

- Characterize the trees with $a_u(T) = 1$.
(We think we know which caterpillars work.)

- What is the complexity of recognizing $a_u(G) = 1$?

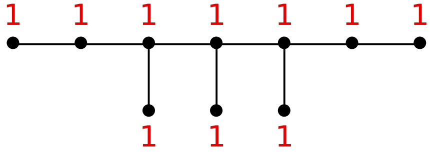
Fractional Acquisition

- Always $a_f(G) \le a_u(G) \le a_t(G)$, but $a_f(P_n) = a_u(P_n) = a_t(P_n) = \lceil n/4 \rceil$. (Same values for C_n.)

Fractional Acquisition

- Always $a_f(G) \le a_u(G) \le a_t(G)$, but $a_f(P_n) = a_u(P_n) = a_t(P_n) = \lceil n/4 \rceil$. (Same values for C_n.)

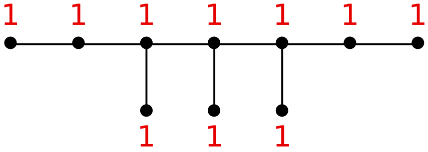
Ex. Fractional moves may help: In the graph below, $a_u(G) = a_t(G) = 2$ (no ascending tree can be made).



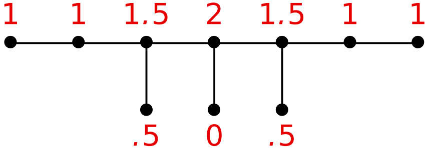
Fractional Acquisition

- Always $a_f(G) \le a_u(G) \le a_t(G)$, but $a_f(P_n) = a_u(P_n) = a_t(P_n) = \lceil n/4 \rceil$. (Same values for C_n.)

Ex. Fractional moves may help: In the graph below, $a_u(G) = a_t(G) = 2$ (no ascending tree can be made).



Fractional moves create an ascending tree: $a_f(G) = 1$.



Surprise!

- If G is a path or cycle, then $\alpha_f(G) = \lceil n/4 \rceil$.

Surprise!

- If G is a path or cycle, then $\alpha_f(G) = \lceil n/4 \rceil$.
- Otherwise, $\Delta(G) \geq 3$.

Surprise!

- If G is a path or cycle, then $\alpha_f(G) = \lceil n/4 \rceil$.
- Otherwise, $\Delta(G) \geq 3$.

Thm. (Wenger) $\Delta(G) \geq 3$ and connected

Surprise!

- If G is a path or cycle, then $\alpha_f(G) = \lceil n/4 \rceil$.
- Otherwise, $\Delta(G) \geq 3$.

Thm. (Wenger) $\Delta(G) \geq 3$ and connected $\Rightarrow \alpha_f(G) = 1$.

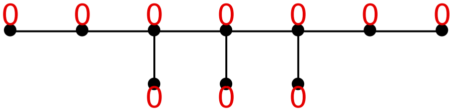
Surprise!

- If G is a path or cycle, then $\alpha_f(G) = \lceil n/4 \rceil$.
- Otherwise, $\Delta(G) \geq 3$.

Thm. (Wenger) $\Delta(G) \geq 3$ and connected $\Rightarrow \alpha_f(G) = 1$.

Main Steps:

1) New model: make all initial weights 0, require integer moves, and allow negative weights.



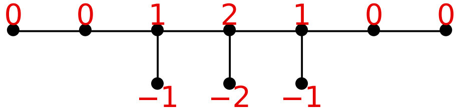
Surprise!

- If G is a path or cycle, then $\alpha_f(G) = \lceil n/4 \rceil$.
- Otherwise, $\Delta(G) \geq 3$.

Thm. (Wenger) $\Delta(G) \geq 3$ and connected $\Rightarrow \alpha_f(G) = 1$.

Main Steps:

1) New model: make all initial weights 0, require integer moves, and allow negative weights.



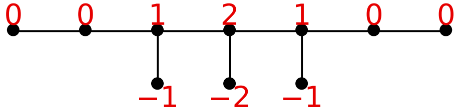
Surprise!

- If G is a path or cycle, then $a_f(G) = \lceil n/4 \rceil$.
- Otherwise, $\Delta(G) \geq 3$.

Thm. (Wenger) $\Delta(G) \geq 3$ and connected $\Rightarrow a_f(G) = 1$.

Main Steps:

1) New model: make all initial weights 0, require integer moves, and allow negative weights.



2) An ascending tree can be normalized: divide by the largest magnitude of weight ever used in the process and add 1. The corresponding fractional moves in the original problem produce an ascending tree.

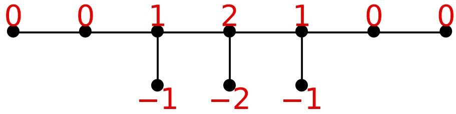
Surprise!

- If G is a path or cycle, then $a_f(G) = \lceil n/4 \rceil$.
- Otherwise, $\Delta(G) \geq 3$.

Thm. (Wenger) $\Delta(G) \geq 3$ and connected $\Rightarrow a_f(G) = 1$.

Main Steps:

1) New model: make all initial weights 0, require integer moves, and allow negative weights.



2) An ascending tree can be normalized: divide by the largest magnitude of weight ever used in the process and add 1. The corresponding fractional moves in the original problem produce an ascending tree.

3) Inductively produce an ascending tree in this model.

Game Acquisition

Def. (Slater–Wang [2004]) Min and Max alternate total acquisition moves, aiming to minimize or maximize the final set. The game acquisition number $\alpha_g(G)$ is the result when Min starts ($\hat{\alpha}_g(G)$ when Max starts).

Game Acquisition

Def. (Slater–Wang [2004]) Min and Max alternate total acquisition moves, aiming to minimize or maximize the final set. The game acquisition number $a_g(G)$ is the result when Min starts ($\hat{a}_g(G)$ when Max starts).

Thm. (Slater–Wang [2004] $a_g(P_n) = 2n/5$.

Game Acquisition

Def. (Slater–Wang [2004]) Min and Max alternate total acquisition moves, aiming to minimize or maximize the final set. The game acquisition number $a_g(G)$ is the result when Min starts ($\hat{a}_g(G)$ when Max starts).

Thm. (Slater–Wang [2004] $a_g(P_n) = 2n/5$.

Ex. Who moves first?: $a_g(K_{1,q}) = 1$, but $\hat{a}_g(K_{1,q}) = q$.

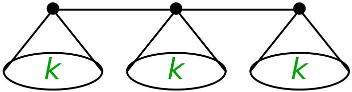
Game Acquisition

Def. (Slater–Wang [2004]) Min and Max alternate total acquisition moves, aiming to minimize or maximize the final set. The game acquisition number $a_g(G)$ is the result when Min starts ($\hat{a}_g(G)$ when Max starts).

Thm. (Slater–Wang [2004] $a_g(P_n) = 2n/5$.

Ex. Who moves first?: $a_g(K_{1,q}) = 1$, but $\hat{a}_g(K_{1,q}) = q$.

Ex. For the tree T below, $a_g(T) \approx 2n/3$.
Max first kills one end of the spine and combines the two remaining spine vertices in the second round.



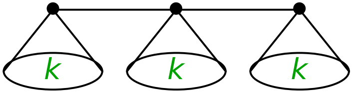
Game Acquisition

Def. (Slater–Wang [2004]) Min and Max alternate total acquisition moves, aiming to minimize or maximize the final set. The game acquisition number $a_g(G)$ is the result when Min starts ($\hat{a}_g(G)$ when Max starts).

Thm. (Slater–Wang [2004] $a_g(P_n) = 2n/5$.

Ex. Who moves first?: $a_g(K_{1,q}) = 1$, but $\hat{a}_g(K_{1,q}) = q$.

Ex. For the tree T below, $a_g(T) \approx 2n/3$.
Max first kills one end of the spine and combines the two remaining spine vertices in the second round.



Ques. What is $\max a_g(T)$ among n-vertex trees?

Results for $K_{m,n}$ (MMSWW)

Fix $1 \leq m \leq n$, partite sets X, Y with $|X| = m$, $|Y| = n$.

Results for $K_{m,n}$ (MMSWW)

Fix $1 \le m \le n$, partite sets X, Y with $|X| = m$, $|Y| = n$.

Upper bd = Min strategy; Lower bd = Max strategy.

Results for $K_{m,n}$ (MMSWW)

Fix $1 \le m \le n$, partite sets X, Y with $|X| = m$, $|Y| = n$.

Upper bd = Min strategy; Lower bd = Max strategy.

Thm. $\hat{a}_g(K_{m,n}) = n - m + 1$. (Max-start)

Results for $K_{m,n}$ (MMSWW)

Fix $1 \le m \le n$, partite sets X, Y with $|X| = m$, $|Y| = n$.

Upper bd = Min strategy; Lower bd = Max strategy.

Thm. $\hat{\alpha}_g(K_{m,n}) = n - m + 1$. (Max-start)

Thm. $\alpha_g(K_{m,n}) \le \left\lfloor \frac{n-m}{3} \right\rfloor + 2$. (Min strategy)

Results for $K_{m,n}$ (MMSWW)

Fix $1 \leq m \leq n$, partite sets X, Y with $|X| = m$, $|Y| = n$.

Upper bd = Min strategy; Lower bd = Max strategy.

Thm. $\hat{\alpha}_g(K_{m,n}) = n - m + 1$. (Max-start)

Thm. $\alpha_g(K_{m,n}) \leq \left\lfloor \frac{n-m}{3} \right\rfloor + 2$. (Min strategy)

(Equality for $n - m$ small; in particular, $\alpha_g(K_{n,n}) = 2$.)

Results for $K_{m,n}$ (MMSWW)

Fix $1 \le m \le n$, partite sets X, Y with $|X| = m$, $|Y| = n$.

Upper bd = Min strategy; Lower bd = Max strategy.

Thm. $\hat{\alpha}_g(K_{m,n}) = n - m + 1$. (Max-start)

Thm. $\alpha_g(K_{m,n}) \le \left\lfloor \frac{n-m}{3} \right\rfloor + 2$. (Min strategy)

(Equality for $n - m$ small; in particular, $\alpha_g(K_{n,n}) = 2$.)

Thm. $\alpha_g(K_{m,n}) \le 2\log_{3/2} m + 18$. (Min strategy)

Results for $K_{m,n}$ (MMSWW)

Fix $1 \le m \le n$, partite sets X, Y with $|X| = m$, $|Y| = n$.

Upper bd = Min strategy; Lower bd = Max strategy.

Thm. $\hat{\alpha}_g(K_{m,n}) = n - m + 1$. (Max-start)

Thm. $\alpha_g(K_{m,n}) \le \left\lfloor \frac{n-m}{3} \right\rfloor + 2$. (Min strategy)

(Equality for $n - m$ small; in particular, $\alpha_g(K_{n,n}) = 2$.)

Thm. $\alpha_g(K_{m,n}) \le 2\log_{3/2} m + 18$. (Min strategy)

(Max ensures at least $\log_2 m - c$ for $n - m$ large.)

Results for $K_{m,n}$ (MMSWW)

Fix $1 \le m \le n$, partite sets X, Y with $|X| = m$, $|Y| = n$.

Upper bd = Min strategy; Lower bd = Max strategy.

Thm. $\hat{a}_g(K_{m,n}) = n - m + 1$. (Max-start)

Thm. $a_g(K_{m,n}) \le \left\lfloor \frac{n-m}{3} \right\rfloor + 2$. (Min strategy)

(Equality for $n - m$ small; in particular, $a_g(K_{n,n}) = 2$.)

Thm. $a_g(K_{m,n}) \le 2\log_{3/2} m + 18$. (Min strategy)

(Max ensures at least $\log_2 m - c$ for $n - m$ large.)

Ques. Note that $a_g(K_{n,n}) = 2$ but $\hat{a}_g(K_{n,n}) = 1$.

Results for $K_{m,n}$ (MMSWW)

Fix $1 \le m \le n$, partite sets X, Y with $|X| = m$, $|Y| = n$.

Upper bd = Min strategy; Lower bd = Max strategy.

Thm. $\hat{a}_g(K_{m,n}) = n - m + 1$. (Max-start)

Thm. $a_g(K_{m,n}) \le \left\lfloor \frac{n-m}{3} \right\rfloor + 2$. (Min strategy)

(Equality for $n - m$ small; in particular, $a_g(K_{n,n}) = 2$.)

Thm. $a_g(K_{m,n}) \le 2 \log_{3/2} m + 18$. (Min strategy)

(Max ensures at least $\log_2 m - c$ for $n - m$ large.)

Ques. Note that $a_g(K_{n,n}) = 2$ but $\hat{a}_g(K_{n,n}) = 1$.

When is $a_g(G) - \hat{a}_g(G)$ positive? How big can it be?

A Strategy for Max

Live vertices are pawns (wt $= 1$) or kings (wt > 1).

A Strategy for Max

Live vertices are pawns (wt $= 1$) or kings (wt > 1).

Lem. Suppose X has p pawns (only) and Y has s pawns and t kings, with $t \geq 1$ and $s \geq p$. If Min moves next, then Max can ensure $\geq s+t-p$ live vertices at the end.

A Strategy for Max

Live vertices are pawns (wt $= 1$) or kings (wt > 1).

Lem. Suppose X has p pawns (only) and Y has s pawns and t kings, with $t \geq 1$ and $s \geq p$. If Min moves next, then Max can ensure $\geq s+t-p$ live vertices at the end.

Pf. Induction on p. If $p = 0$, the game is over.

A Strategy for Max

Live vertices are pawns (wt $= 1$) or kings (wt > 1).

Lem. Suppose X has p pawns (only) and Y has s pawns and t kings, with $t \geq 1$ and $s \geq p$. If Min moves next, then Max can ensure $\geq s+t-p$ live vertices at the end.

Pf. Induction on p. If $p = 0$, the game is over.

For $p > 0$, note that X has no king.

A Strategy for Max

Live vertices are pawns (wt $=1$) or kings (wt >1).

Lem. Suppose X has p pawns (only) and Y has s pawns and t kings, with $t \geq 1$ and $s \geq p$. If Min moves next, then Max can ensure $\geq s+t-p$ live vertices at the end.

Pf. Induction on p. If $p=0$, the game is over.

For $p>0$, note that X has no king.

(1) If Min makes a king in X, then Max absorbs it into Y.

(2) If Min absorbs a pawn into Y, then Max does also.

A Strategy for Max

Live vertices are pawns (wt $= 1$) or kings (wt > 1).

Lem. Suppose X has p pawns (only) and Y has s pawns and t kings, with $t \geq 1$ and $s \geq p$. If Min moves next, then Max can ensure $\geq s+t-p$ live vertices at the end.

Pf. Induction on p. If $p = 0$, the game is over.

For $p > 0$, note that X has no king.

(1) If Min makes a king in X, then Max absorbs it into Y.

(2) If Min absorbs a pawn into Y, then Max does also.

A round leaves no king in X, reduces p, and does not reduce t or $s - p$. The induction hypothesis applies. ■

A Strategy for Max

Live vertices are pawns (wt $= 1$) or kings (wt > 1).

Lem. Suppose X has p pawns (only) and Y has s pawns and t kings, with $t \geq 1$ and $s \geq p$. If Min moves next, then Max can ensure $\geq s+t-p$ live vertices at the end.

Pf. Induction on p. If $p = 0$, the game is over.

For $p > 0$, note that X has no king.

(1) If Min makes a king in X, then Max absorbs it into Y.

(2) If Min absorbs a pawn into Y, then Max does also.

A round leaves no king in X, reduces p, and does not reduce t or $s - p$. The induction hypothesis applies. ∎

Thm. $\hat{a}_g(K_{m,n}) \geq n - m + 1$.

A Strategy for Max

Live vertices are pawns (wt $= 1$) or kings (wt > 1).

Lem. Suppose X has p pawns (only) and Y has s pawns and t kings, with $t \geq 1$ and $s \geq p$. If Min moves next, then Max can ensure $\geq s+t-p$ live vertices at the end.

Pf. Induction on p. If $p = 0$, the game is over.

For $p > 0$, note that X has no king.

(1) If Min makes a king in X, then Max absorbs it into Y.

(2) If Min absorbs a pawn into Y, then Max does also.

A round leaves no king in X, reduces p, and does not reduce t or $s - p$. The induction hypothesis applies. ■

Thm. $\hat{a}_g(K_{m,n}) \geq n - m + 1$.

Pf. Max makes a king in Y: $t = 1$, $p = m - 1$, $s = n - 1$. ■

A Strategy for Min

Lem. Given: p pawns and q kings in X, s pawns and t kings in Y, $q, t \geq 1$ and $s \geq p$. Let $r = \max\{0, s - p - q + 1\}$. If Max moves next, then Min ensures ending with $\leq \max\{q, t+r\}$ live vertices.

A Strategy for Min

Lem. Given: p pawns and q kings in X, s pawns and t kings in Y, $q,t \geq 1$ and $s \geq p$. Let $r = \max\{0, s-p-q+1\}$. If Max moves next, then Min ensures ending with $\leq \max\{q, t+r\}$ live vertices.

Pf. Induction on p. For $p = 0$, Min absorbs pawns from Y into kings in X. Game ends in X with $\leq q$ kings or in Y with $\leq t$ kings and $\leq r$ pawns.

A Strategy for Min

Lem. Given: p pawns and q kings in X, s pawns and t kings in Y, $q, t \geq 1$ and $s \geq p$. Let $r = \max\{0, s - p - q + 1\}$. If Max moves next, then Min ensures ending with $\leq \max\{q, t+r\}$ live vertices.

Pf. Induction on p. For $p = 0$, Min absorbs pawns from Y into kings in X. Game ends in X with $\leq q$ kings or in Y with $\leq t$ kings and $\leq r$ pawns. For $p > 0$, Min plays:

A Strategy for Min

Lem. Given: p pawns and q kings in X,
s pawns and t kings in Y, $q, t \geq 1$ and $s \geq p$.
Let $r = \max\{0, s - p - q + 1\}$. If Max moves next, then Min ensures ending with $\leq \max\{q, t+r\}$ live vertices.

Pf. Induction on p. For $p = 0$, Min absorbs pawns from Y into kings in X. Game ends in X with $\leq q$ kings or in Y with $\leq t$ kings and $\leq r$ pawns. For $p > 0$, Min plays:

(1) If Max creates a king, then Min absorbs it.

(2) If Max absorbs a king, then Min replaces it.

(3) If Max absorbs a pawn by a king, then Min does also.

A Strategy for Min

Lem. Given: p pawns and q kings in X,
s pawns and t kings in Y, $q, t \geq 1$ and $s \geq p$.
Let $r = \max\{0, s - p - q + 1\}$. If Max moves next, then Min ensures ending with $\leq \max\{q, t + r\}$ live vertices.

Pf. Induction on p. For $p = 0$, Min absorbs pawns from Y into kings in X. Game ends in X with $\leq q$ kings or in Y with $\leq t$ kings and $\leq r$ pawns. For $p > 0$, Min plays:

(1) If Max creates a king, then Min absorbs it.

(2) If Max absorbs a king, then Min replaces it.

(3) If Max absorbs a pawn by a king, then Min does also.

A round reduces p, s by 1 and does not change q, t. ■

A Strategy for Min

Lem. Given: p pawns and q kings in X, s pawns and t kings in Y, $q, t \geq 1$ and $s \geq p$. Let $r = \max\{0, s - p - q + 1\}$. If Max moves next, then Min ensures ending with $\leq \max\{q, t+r\}$ live vertices.

Pf. Induction on p. For $p = 0$, Min absorbs pawns from Y into kings in X. Game ends in X with $\leq q$ kings or in Y with $\leq t$ kings and $\leq r$ pawns. For $p > 0$, Min plays:

(1) If Max creates a king, then Min absorbs it.

(2) If Max absorbs a king, then Min replaces it.

(3) If Max absorbs a pawn by a king, then Min does also.

A round reduces p, s by 1 and does not change q, t. ■

Thm. $\hat{a}_g(K_{m,n}) \leq n - m + 1$.

A Strategy for Min

Lem. Given: p pawns and q kings in X, s pawns and t kings in Y, $q,t \geq 1$ and $s \geq p$. Let $r = \max\{0, s-p-q+1\}$. If Max moves next, then Min ensures ending with $\leq \max\{q, t+r\}$ live vertices.

Pf. Induction on p. For $p=0$, Min absorbs pawns from Y into kings in X. Game ends in X with $\leq q$ kings or in Y with $\leq t$ kings and $\leq r$ pawns. For $p>0$, Min plays:

(1) If Max creates a king, then Min absorbs it.

(2) If Max absorbs a king, then Min replaces it.

(3) If Max absorbs a pawn by a king, then Min does also.

A round reduces p,s by 1 and does not change q,t. ■

Thm. $\hat{a}_g(K_{m,n}) \leq n-m+1$.

Pf. Max initially makes a king; Min makes a king on the other side. Now $q=t=1$, $p=m-2$, $s=n-2$. ■

Min-Start: The Balanced Case

Thm. $a_g(K_{n,n}) = 2$.

Min-Start: The Balanced Case

Thm. $a_g(K_{n,n}) = 2$.

Pf. Min initially makes a king.

Min-Start: The Balanced Case

Thm. $a_g(K_{n,n}) = 2$.

Pf. Min initially makes a king.

Lower Bound: Max makes a king on the same side.
The Max-strategy Lemma applies:
$s + t - p = (n-2) + 2 - (n-2)$.

Min-Start: The Balanced Case

Thm. $a_g(K_{n,n}) = 2$.

Pf. Min initially makes a king.

Lower Bound: Max makes a king on the same side. The Max-strategy Lemma applies: $s+t-p = (n-2)+2-(n-2)$.

Upper Bound: Max makes another king or absorbs a pawn. Min ensures that each side has a king. Applying the Min-strategy Lemma, $\max\{q, t+(s-p-q+1)\}$ is $\max\{2, 1+(-2+1)\}$ or $\max\{1, 1+(1-1+1)\}$. ■

Min-Start: The Balanced Case

Thm. $a_g(K_{n,n}) = 2$.

Pf. Min initially makes a king.

Lower Bound: Max makes a king on the same side. The Max-strategy Lemma applies: $s+t-p = (n-2)+2-(n-2)$.

Upper Bound: Max makes another king or absorbs a pawn. Min ensures that each side has a king. Applying the Min-strategy Lemma, $\max\{q, t+(s-p-q+1)\}$ is $\max\{2, 1+(-2+1)\}$ or $\max\{1, 1+(1-1+1)\}$. ■

Idea of general upper bound:
After first making a king in X, Min can absorb any kings made by Max in Y to achieve $a_g(K_{m,n}) \leq m$.

Min-Start: The Balanced Case

Thm. $a_g(K_{n,n}) = 2$.

Pf. Min initially makes a king.

Lower Bound: Max makes a king on the same side. The Max-strategy Lemma applies: $s+t-p = (n-2)+2-(n-2)$.

Upper Bound: Max makes another king or absorbs a pawn. Min ensures that each side has a king. Applying the Min-strategy Lemma, $\max\{q, t+(s-p-q+1)\}$ is $\max\{2, 1+(-2+1)\}$ or $\max\{1, 1+(1-1+1)\}$. ∎

Idea of general upper bound:
After first making a king in X, Min can absorb any kings made by Max in Y to achieve $a_g(K_{m,n}) \le m$.

Better: Min creates q kings in X to employ the bound $\max\{q, t+(s-p-q+1)\}$ in the Min-strategy Lemma.

Min-Start: Upper Bound when $n - m < 3m$

Thm. $a_g(K_{m,n}) \leq \left\lfloor \frac{n-m}{3} \right\rfloor + 2$.

Min-Start: Upper Bound when $n - m < 3m$

Thm. $a_g(K_{m,n}) \le \left\lfloor \frac{n-m}{3} \right\rfloor + 2$.

Pf. Min first makes a king in X. While X has at most $\left\lceil \frac{n-m}{3} \right\rceil$ kings, Min plays this, never leaving a king in Y:

Min-Start: Upper Bound when $n - m < 3m$

Thm. $a_g(K_{m,n}) \le \left\lfloor \frac{n-m}{3} \right\rfloor + 2$.

Pf. Min first makes a king in X. While X has at most $\left\lceil \frac{n-m}{3} \right\rceil$ kings, Min plays this, never leaving a king in Y:

(1) If Max makes a king in Y, then Min absorbs it to X.

(2) If Max makes a king in X, then Min adds a pawn to it.

(3) If Max adds a pawn from Y to a king in X, then Min makes another king in X.

Min-Start: Upper Bound when $n-m<3m$

Thm. $a_g(K_{m,n}) \le \left\lfloor \frac{n-m}{3} \right\rfloor + 2$.

Pf. Min first makes a king in X. While X has at most $\left\lceil \frac{n-m}{3} \right\rceil$ kings, Min plays this, never leaving a king in Y:

(1) If Max makes a king in Y, then Min absorbs it to X.

(2) If Max makes a king in X, then Min adds a pawn to it.

(3) If Max adds a pawn from Y to a king in X, then Min makes another king in X.

Each round takes one pawn from each side, plus an extra pawn from Y for each king made in X.

Min-Start: Upper Bound when $n - m < 3m$

Thm. $a_g(K_{m,n}) \le \left\lfloor \frac{n-m}{3} \right\rfloor + 2$.

Pf. Min first makes a king in X. While X has at most $\left\lceil \frac{n-m}{3} \right\rceil$ kings, Min plays this, never leaving a king in Y:

(1) If Max makes a king in Y, then Min absorbs it to X.

(2) If Max makes a king in X, then Min adds a pawn to it.

(3) If Max adds a pawn from Y to a king in X, then Min makes another king in X.

Each round takes one pawn from each side, plus an extra pawn from Y for each king made in X.

When X has $\left\lceil \frac{n-m}{3} \right\rceil + 1$ kings, Min makes a king in Y. With Max to move, the Min-strategy Lemma applies with $q = \left\lceil \frac{n-m}{3} \right\rceil + 1$, $t = 1$, and $s - p \le (n-m) - (q-2)$.

Min-Start: Upper Bound when $n-m<3m$

Thm. $a_g(K_{m,n}) \le \left\lfloor \frac{n-m}{3} \right\rfloor + 2$.

Pf. Min first makes a king in X. While X has at most $\left\lceil \frac{n-m}{3} \right\rceil$ kings, Min plays this, never leaving a king in Y:

(1) If Max makes a king in Y, then Min absorbs it to X.

(2) If Max makes a king in X, then Min adds a pawn to it.

(3) If Max adds a pawn from Y to a king in X, then Min makes another king in X.

Each round takes one pawn from each side, plus an extra pawn from Y for each king made in X.

When X has $\left\lceil \frac{n-m}{3} \right\rceil + 1$ kings, Min makes a king in Y. With Max to move, the Min-strategy Lemma applies with $q = \left\lceil \frac{n-m}{3} \right\rceil + 1$, $t = 1$, and $s-p \le (n-m)-(q-2)$.

Thus Min ensures at most $\max\{q, t+(n-m)-2q+3\}$ live vertices, i.e., at most $\left\lfloor \frac{n-m}{3} \right\rfloor + 2$. ■

Min-Start: Upper Bound for Large $n - m$

Idea: Min introduces a temporary king into Y to absorb some kings from X, making sure it does not get heavy.

Min-Start: Upper Bound for Large $n - m$

Idea: Min introduces a temporary king into Y to absorb some kings from X, making sure it does not get heavy.

Def. $\hat{x}$ and $\hat{y}$ are currently heaviest vertices in X and Y.
$w(v)$ is the current weight of v.

Min-Start: Upper Bound for Large $n - m$

Idea: Min introduces a temporary king into Y to absorb some kings from X, making sure it does not get heavy.

Def. $\hat{x}$ and $\hat{y}$ are currently heaviest vertices in X and Y.
$w(v)$ is the current weight of v.
A Min move is safe if it leaves $w(\hat{x}) \geq 2w(\hat{y})$ and at most one king in Y. (The initial Min move is safe.)

Min-Start: Upper Bound for Large $n - m$

Idea: Min introduces a temporary king into Y to absorb some kings from X, making sure it does not get heavy.

Def. $\hat{x}$ and $\hat{y}$ are currently heaviest vertices in X and Y.
$w(v)$ is the current weight of v.
A Min move is safe if it leaves $w(\hat{x}) \geq 2w(\hat{y})$ and at most one king in Y. (The initial Min move is safe.)

Algorithm: (until Y has no king and X has no pawn)
(1) If Max makes a king in Y **or** $w(\hat{x}) < 2(w(\hat{y}) + 2)$,
then Min absorbs $\hat{y}$ into $\hat{x}$.
(2) If Max doesn't make king in Y **and** $w(\hat{x}) \geq 2(w(\hat{y})+2)$,
then Min absorbs into $\hat{y}$ a king of weight 2 or a pawn.

Min-Start: Upper Bound for Large $n - m$

Idea: Min introduces a temporary king into Y to absorb some kings from X, making sure it does not get heavy.

Def. $\hat{x}$ and $\hat{y}$ are currently heaviest vertices in X and Y.
$w(v)$ is the current weight of v.
A Min move is safe if it leaves $w(\hat{x}) \geq 2w(\hat{y})$ and at most one king in Y. (The initial Min move is safe.)

Algorithm: (until Y has no king and X has no pawn)
(1) If Max makes a king in Y **or** $w(\hat{x}) < 2(w(\hat{y}) + 2)$,
then Min absorbs $\hat{y}$ into $\hat{x}$.
(2) If Max doesn't make king in Y **and** $w(\hat{x}) \geq 2(w(\hat{y}) + 2)$,
then Min absorbs into $\hat{y}$ a king of weight 2 or a pawn.

Thm. $a_g(K_{m,n}) \leq 2 \log_{3/2} m + 18$.

Min-Start: Upper Bound for Large $n - m$

Def. $\hat{x}$ and $\hat{y}$ are currently heaviest vertices in X and Y.
$w(v)$ is the current weight of v.

A Min move is safe if it leaves $w(\hat{x}) \geq 2w(\hat{y})$ and at most one king in Y. (The initial Min move is safe.)

Algorithm: (until Y has no king and X has no pawn)

(1) If Max makes a king in Y **or** $w(\hat{x}) < 2(w(\hat{y}) + 2)$,
then Min absorbs $\hat{y}$ into $\hat{x}$.

(2) If Max doesn't make king in Y **and** $w(\hat{x}) \geq 2(w(\hat{y})+2)$,
then Min absorbs into $\hat{y}$ a king of weight 2 or a pawn.

Thm. $a_g(K_{m,n}) \leq 2 \log_{3/2} m + 18$.

Step 1: Min moves are safe, and the game ends in X.

Min-Start: Upper Bound for Large $n - m$

Def. $\hat{x}$ and $\hat{y}$ are currently heaviest vertices in X and Y.
$w(v)$ is the current weight of v.

A Min move is safe if it leaves $w(\hat{x}) \geq 2w(\hat{y})$ and at most one king in Y. (The initial Min move is safe.)

Algorithm: (until Y has no king and X has no pawn)

(1) If Max makes a king in Y **or** $w(\hat{x}) < 2(w(\hat{y}) + 2)$, then Min absorbs $\hat{y}$ into $\hat{x}$.

(2) If Max doesn't make king in Y **and** $w(\hat{x}) \geq 2(w(\hat{y}) + 2)$, then Min absorbs into $\hat{y}$ a king of weight 2 or a pawn.

Thm. $a_g(K_{m,n}) \leq 2 \log_{3/2} m + 18$.

Step 1: Min moves are safe, and the game ends in X.

Step 2: $q + \max\{0, p - s\}$ starts at 1, increases by at most 2 for each Type 1 move, and ends at $|X|$.

Min-Start: Upper Bound for Large $n-m$

Def. $\hat{x}$ and $\hat{y}$ are currently heaviest vertices in X and Y.
$w(v)$ is the current weight of v.
A Min move is safe if it leaves $w(\hat{x}) \ge 2w(\hat{y})$ and at most one king in Y. (The initial Min move is safe.)

Algorithm: (until Y has no king and X has no pawn)
(1) If Max makes a king in Y **or** $w(\hat{x}) < 2(w(\hat{y})+2)$, then Min absorbs $\hat{y}$ into $\hat{x}$.
(2) If Max doesn't make king in Y **and** $w(\hat{x}) \ge 2(w(\hat{y})+2)$, then Min absorbs into $\hat{y}$ a king of weight 2 or a pawn.

Thm. $a_g(K_{m,n}) \le 2\log_{3/2} m + 18$.

Step 1: Min moves are safe, and the game ends in X.

Step 2: $q + \max\{0, p-s\}$ starts at 1, increases by at most 2 for each Type 1 move, and ends at $|X|$.

Step 3: Each Type 1 move increases $w(\hat{x})$ by a factor of at least $3/2$, and $w(\hat{x})$ cannot exceed $6m$.

Min-Start: Lower Bound

Idea: Max makes medium-weight kings in X.

Min-Start: Lower Bound

Idea: Max makes medium-weight kings in X.

This keeps Min from absorbing them into Y and then X.

Min-Start: Lower Bound

Idea: Max makes medium-weight kings in X.

This keeps Min from absorbing them into Y and then X.

It also threatens to make $\hat{y}$ heavier than $\hat{x}$,
which would end the game in Y.

Min-Start: Lower Bound

Idea: Max makes medium-weight kings in X.

This keeps Min from absorbing them into Y and then X.

It also threatens to make $\hat{y}$ heavier than $\hat{x}$,
which would end the game in Y.

Def. At a given time, let
$X^{-} = \{x \in X\colon\ 0 < w(x) \leq \frac{1}{2} w(\hat{x})\}$,
x' = a heaviest in X with $w(x') \leq w(\hat{y})$,
y' = a heaviest in Y with $w(y') \leq w(x)$ for some $x \in X^{-}$,
x^{*} = a lightest in X with $w(x^{*}) \geq w(y')$.

Min-Start: Lower Bound

Idea: Max makes medium-weight kings in X.

This keeps Min from absorbing them into Y and then X.

It also threatens to make $\hat{y}$ heavier than $\hat{x}$,
which would end the game in Y.

Def. At a given time, let
$X^- = \{x \in X : 0 < w(x) \le \frac{1}{2} w(\hat{x})\}$,
x' = a heaviest in X with $w(x') \le w(\hat{y})$,
y' = a heaviest in Y with $w(y') \le w(x)$ for some $x \in X^-$,
x^* = a lightest in X with $w(x^*) \ge w(y')$.

While the vertices x', y', x^* exist, Max plays as follows:

Min-Start: Lower Bound

Idea: Max makes medium-weight kings in X.

This keeps Min from absorbing them into Y and then X.

It also threatens to make $\hat{y}$ heavier than $\hat{x}$,
which would end the game in Y.

Def. At a given time, let
$X^- = \{x \in X\colon\ 0 < w(x) \le \frac{1}{2} w(\hat{x})\}$,
x' = a heaviest in X with $w(x') \le w(\hat{y})$,
y' = a heaviest in Y with $w(y') \le w(x)$ for some $x \in X^-$,
x^* = a lightest in X with $w(x^*) \ge w(y')$.

While the vertices x', y', x^* exist, Max plays as follows:

(1) If $w(x') + w(\hat{y}) \le w(\hat{x})$, then Max absorbs y' into x^*.

(2) If $w(x') + w(\hat{y}) > w(\hat{x})$, then Max absorbs x' into $\hat{y}$.

Min-Start: Lower Bound

Idea: Max makes medium-weight kings in X.

This keeps Min from absorbing them into Y and then X.

It also threatens to make $\hat{y}$ heavier than $\hat{x}$,
which would end the game in Y.

Def. At a given time, let
$X^- = \{x \in X\colon\ 0 < w(x) \leq \frac{1}{2} w(\hat{x})\}$,
x' = a heaviest in X with $w(x') \leq w(\hat{y})$,
y' = a heaviest in Y with $w(y') \leq w(x)$ for some $x \in X^-$,
x^* = a lightest in X with $w(x^*) \geq w(y')$.

While the vertices x', y', x^* exist, Max plays as follows:

(1) If $w(x') + w(\hat{y}) \leq w(\hat{x})$, then Max absorbs y' into x^*.

(2) If $w(x') + w(\hat{y}) > w(\hat{x})$, then Max absorbs x' into $\hat{y}$.

The analysis is difficult!

References

D.E. Lampert and P.J. Slater, The acquisition number of a graph, *Congr. Numer.* 109 (1995), 203–210.

T.D. LeSaulnier and D.B. West, Acquisition-extremal graphs, *Discrete Applied Mathematics* 161 (2013), 1521–1529.

T.D. LeSaulnier, N. Prince, P.S. Wenger, D.B. West, and P. Worah, Total acquisition in graphs, *SIAM J. Discrete Math.* 27 (2013), 1800–1819.

D.C. McDonald, K.G. Milans, C.J. Stocker, D.B. West, and L. Wiglesworth, Game acquisition in graphs, preprint.

N. Prince, P.S. Wenger, and D.B. West, Unit acquisition number, preprint (see Wenger thesis).

P.J. Slater and Y. Wang, The competitive-acquisition numbers of paths, *Congr. Numer.* 167 (2004), 33–43.

P.J. Slater and Y. Wang, Some results on acquisition numbers, *J. Combin. Math. Combin. Comput.* 64 (2008), 65–78.

P.S. Wenger, Fractional acquisition in graphs, submitted.